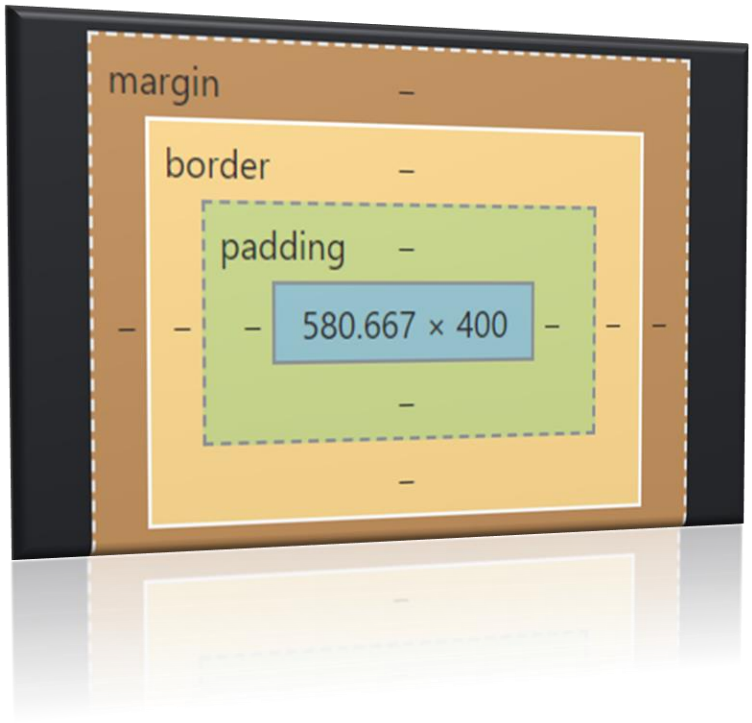
Autor: João Vitor Quirino Sarti
GitHub:https://github.com/NULLBYTE-RGH

CSS Box model



Oque é:

O modelo Box-Model, é o modelo que rege e orquestra tudo que está em uma página web. Tudo em uma página se comporta como uma caixa (podendo ela ser retangular ou quadrada).

Ele é composto por 4 propriedades principais:

- Content
- Padding
- Border
- Margin

Esse modelo é aplicado desde um título <H1> que possui um significado semântico ao HTML até uma <DIV> genérica.

(Uma outra maneira de se referir ao modelo box é como Container. Muito utilizado ao se fazer uso do Bootstrap)

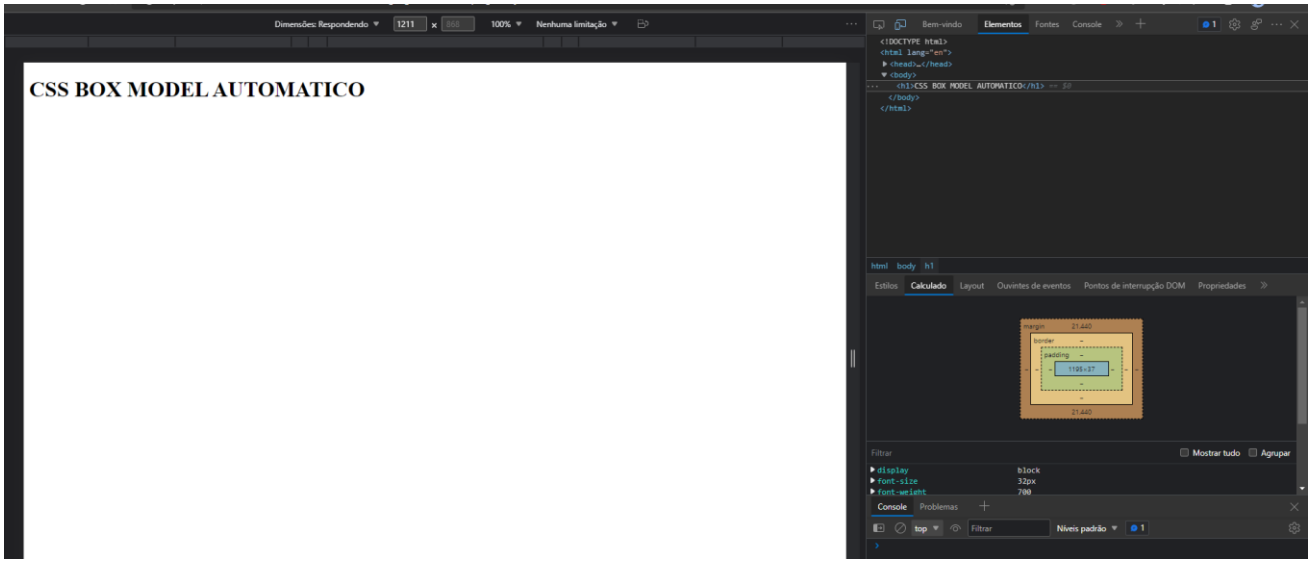
O navegador possui regras CSS pré-carregadas, sendo assim, ao se criar um título por exemplo, e não se colocar as medidas do Box-model, essas regras padrões, serão aplicadas ao conteúdo.

Exemplo:

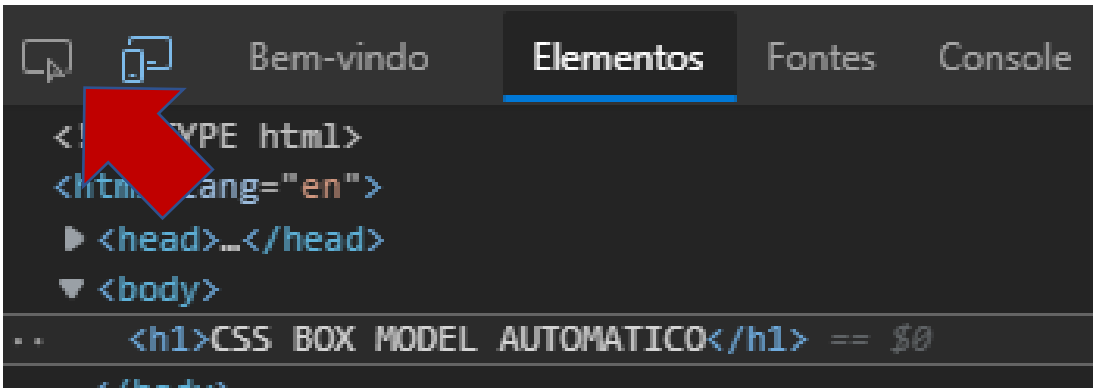
-Criando uma tag H1:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>CSS BOX MODEL AUTOMATICO</h1>
</body>
</html>
```

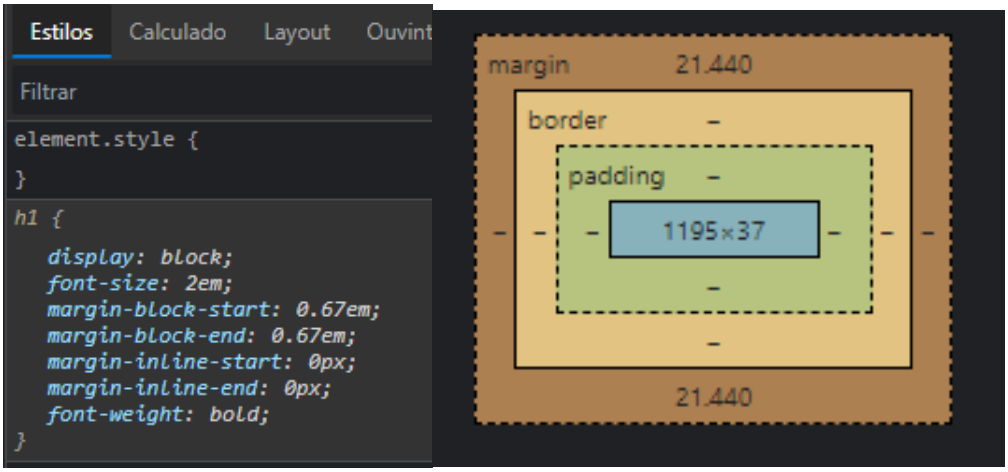
Ao se abrir a página no navegador e executar o inspecionador de elemento com a (tecla F12):



Caso o Box model não apareça, ou caso não seja o box model do elemento que se deseja inspecionar, selecione:



Após isso, basta clicar no texto, no nosso caso, e devera aparecer as seguintes opções:



Assim, já pode ser ver as regras CSS que o navegador aplicou ao elemento.

Ele nos mostra que o display, tipo de exibição, do elemento, usa as propriedades padrões do tipo bloco. Mas a frente será mostrada que existem variações do tipo bloco. Onde essas variações não deixam de ser um bloco, porém com a capacidade de se ter mais de um elemento na mesma linha por exemplo.

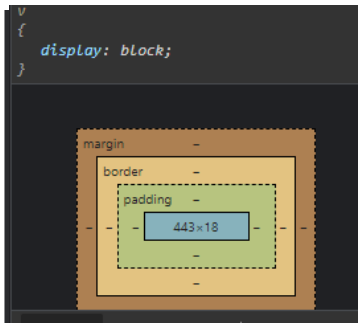
Agora, oque são as outras propriedades? Bom, vamos começar pela Content.

Essa propriedade como o próprio nome diz, é o conteúdo, ou seja, tudo o que definimos dentro da div, podendo ser um texto ou tags filhas.

Vamos fazer um exemplo para ficar mais claro:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    .de{
      color: blue;
      background-color: red;
    }
  </style>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>CSS BOX MODEL AUTOMATICO</h1>
  <div class="de">Eu sou um content!</div>
</body>
</html>
```

Obtemos:



Podemos ver que diferentemente do H1, a caixa de uma DIV, não possui margem definida, isso se dá pelo próprio contexto da DIV ser algo genérico e com o intuito de ser altamente manipulada.

Também há de se notar que não se possui propriedades de margem e padding (preenchimento) definidas até o momento.

Será que é por conta disso que a cor do background foi até o final da página?

Bom, se sua resposta foi SIM, você acertou!

O conteúdo ou content é uma região reservada da caixa, ou seja, a prioridade de espaço é sempre dela. A partir dela que se ditam as regras mais externas, como padding e margin. Mais do que isso: o conteúdo pode extrapolar o tamanho que lhe foi definido no content e até mesmo ultrapassar o padding.

Agora, a pergunta é: como definimos o tamanho desse content?

As 2 propriedades são: width (largura/eixo X) e height (altura/eixo y)

(Essas propriedades se utilizadas com a propriedade padrão box-sizing terá o comportamento de o tamanho definido com o width e height pertencer apenas ao content, mas a frente será apresentada os tipos de box-sizing.)

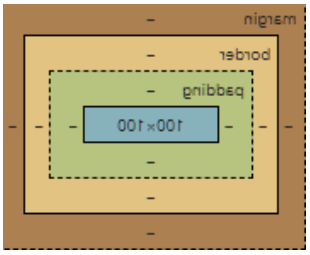
Vamos supor que queiramos fazer nosso texto ocupar um espaço de 100px X 100px

Vamos a um exemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
    .de{
      color: blue;
      background-color: red;
      width: 100px;
      height: 100px;
    }
  </style>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <h1>CSS BOX MODEL AUTOMATICO</h1>
  <div class="de">Eu sou um content!</div>
</body>
</html>
```

CSS BOX MODEL AUTOMATICO

Eu sou um content!

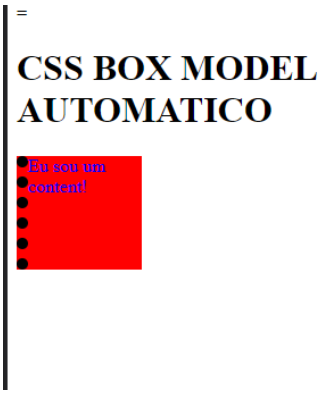


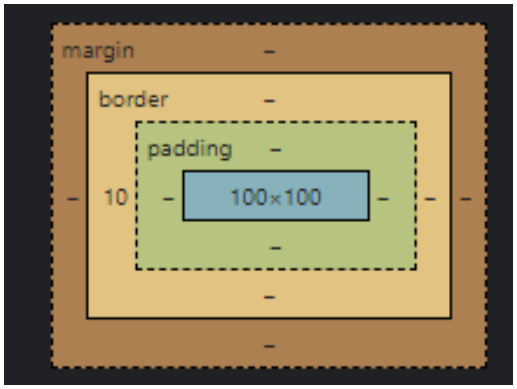
Nesse momento, nosso texto inteiro, junto a cor de fundo ocupam um espaço total de 100x100px, podemos ver que até ocorreu uma quebra de linha automática para o texto se encaixar na largura definida.

Agora, vamos dizer que queremos colocar uma borda na nossa caixa. Para isso utilizaremos as mesmas propriedades vistas na lista 1:

```
border-left: 10px;
border-left-style: dotted;
border-color: black;
```

obtemos:





Isso nos mostra que agora temos mais uma propriedade ativada na nossa box model, e como só adicionamos a borda no lado esquerdo da nossa Box, os 10px só se aplicam a esse pedaço. Porém, como podemos ver, os círculos estão ocupando um espaço grande e estão muito perto do nosso conteúdo.

Agora entra a propriedade Padding (preenchimento), ela vai ser a camada entre a margem e o conteúdo.

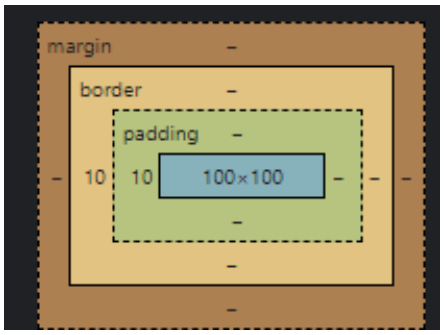
Vamos adicionar a seguinte propriedade a nossa tag style:

```
padding-left: 10px;
```

obtemos:

CSS BOX MODEL AUTOMATICO

Eu sou um content!



(Todas as propriedades mostradas até agora, podem ser aplicadas tanto individualmente como foi feito: padding-left, padding-top etc. Como também representando todos os lados, utilizando simplesmente padding por exemplo)

Agora vamos ativar a última propriedade principal do Box-Model, a margin:

A margem é utilizada para distanciar uma caixa da outra, sendo elas no modelo Block ou Inline Block.

Mas se ainda não definimos nada para se distanciar da outra caixa <H1> por qual razão ele tem uma separação?

Bom essa distância se dá pois como vimos, a tag H1 possui suas propriedades já definidas, e uma delas é a margin. A margem é a distância de uma box a outras caixas, ou até mesmo as bordas da janela do navegador, sendo assim, a margem não só influencia a caixa em que ele foi definida, como todas as caixas próximas a ela.

Para definirmos uma margem, é muito simples; a propriedade é: margin

Vamos adicionar essa propriedade a nossa caixa, e para deixar bem evidente vamos utilizar uma distância de 100px

(Dica: ao se utilizar Pixels como medida seu site não fica responsivo, pois é uma unidade engessada, fixa, para o tornar responsivo utilize por exemplo VW, outras medidas podem ser consultadas nesse site : Guia de Unidades no CSS | Alura)

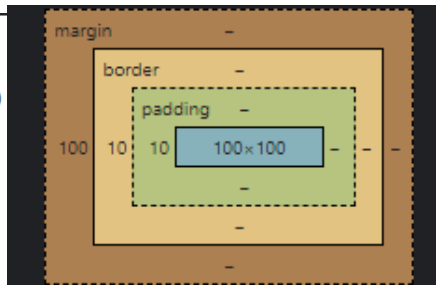
```
<style>
  .de{
    color: blue;
    background-color: red;
    width: 100px;
    height: 100px;
    border-left: 10px;
    border-left-style: dotted;
    border-color: black;
    padding-left: 10px;
    margin-left: 100px;
  }
</style>
```

Resultado:

=

CSS BOX MODEL AUTOMATICO

Eu sou um content!



Muito bem, até agora tudo certo?

Entao.... existe uma outra propriedade da Box-Model, chamada box-sizing. Nós estávamos a utilizando até agora, só que não estávamos definindo-a explicitamente, ou seja, estávamos utilizando o seu valor Default.

Mas no final das contas, o que essa propriedade faz?

A verdade é que ela vai mudar a forma como os tamanhos são calculados.

Lembra que definimos a largura e a altura como 100px X 100px e foi dito que ela pertencia ao tamanho do content? Bom, isso só é verdade se utilizado (padrão/Default):

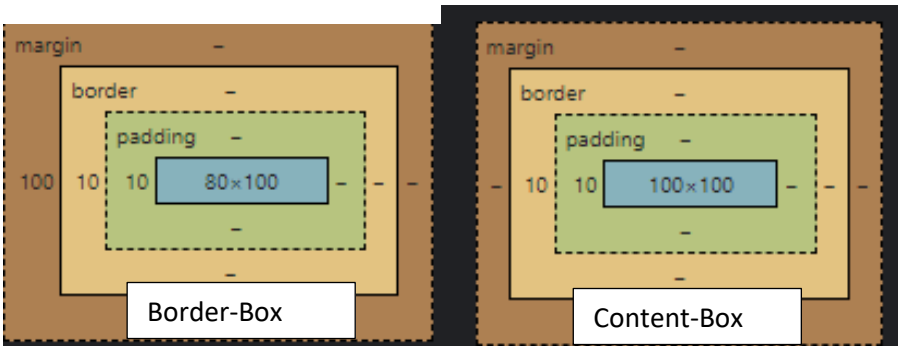
```
box-sizing: content-box;
```

Agora, se utilizarmos a propriedade como:

```
box-sizing: border-box;
```

O tamanho definido no width e height pertencem a caixa toda (content, padding e border).

A partir do momento que for definido como border-box, todas as medidas que passamos serão recalculadas para fazer toda a caixa se encaixar nos novos parâmetros:



Com o Border-Box podemos verificar o tamanho da nossa nova caixa, fazendo a simples soma de tudo no eixo X e tudo no Y:

Em X: Content + padding + border = 80 + 10 + 10 = 100px

Em Y: content + padding + border = 100 +0 + 0 = 100px

Essa propriedade vai ser útil quando se quer ter certeza de que seu elemento da tag, não vá ocupar mais espaço do que o dimensionado.

Extra: (OVERFLOW)

Ao se definir um conteúdo grande, ele pode ultrapassar o tamanho da caixa, causando algo chamado overflow:

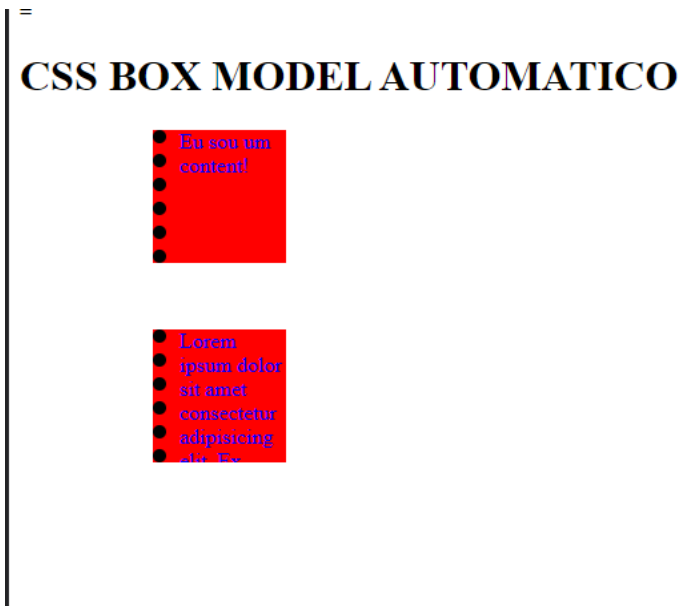
Isso é algo indesejável, e como pode ser resolvido? Bom mais para frente vocês verão outras propriedades CSS que vão lidar muito bem com elementos variáveis e darão a responsividade necessária.

Entao não tem nada que se possa fazer com o Box-Model e display: block? E a resposta é, SIM!

Uma delas é utilizar a propriedade Overflow:

```
OVERFLOW: hidden;
```

Ela simplesmente faz uma corte no conteúdo excedente:



Más pode ser que você não queira que o conteúdo simplesmente sumo e fique dessa forma estranha, como se faltasse uma parte, bom, para isso você pode utilizar a propriedade:

```
OVERFLOW: scroll;
```

Como o próprio nome sugere, você agora tem a opção se usar o scroll do mouse para passar o conteúdo dentro da sua div!

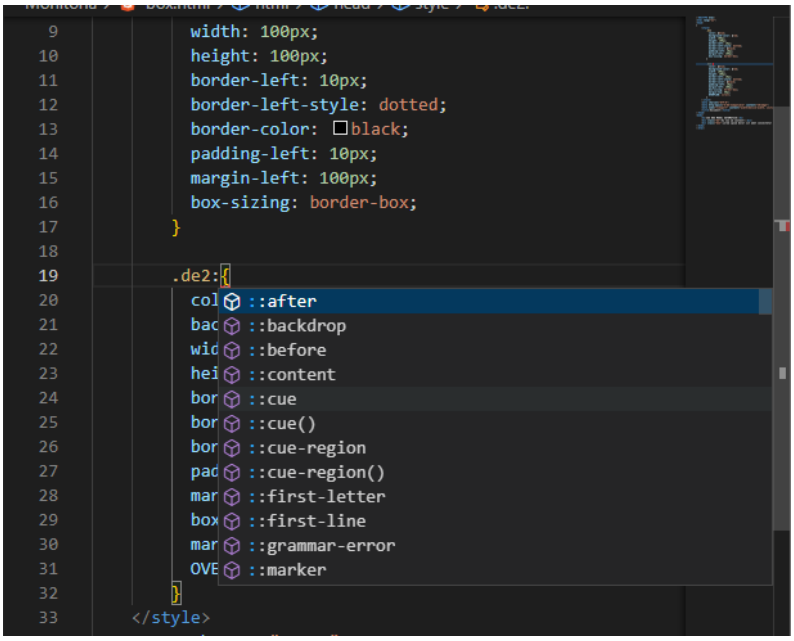
Vou me estender um pouco agora, pois pode ser que nem seja mostrado a vocês isso, mas acho que seja pertinente mostrar.

As propriedades CSS podem ser ativadas por condições, isso sem uso de java-script.

Como?

Dessa forma:

Ao se digitar (:) após uma classe ou ID, o visual code irá sugerir várias propriedades que podem ser utilizadas:



Cada uma dessas propriedades indica que essa classe, no caso de2, será ativada quando um desses eventos ocorrer.

Vou exemplificar com uma propriedade chamada (hover), ele é um evento que indica que o mouse passou em cima da DIV que tem esse evento associado.

E qual a utilidade disso? Responsividade e animação!

Exemplo prático, tomando o Moodle como exemplo; quando o mouse é passado em cima das matérias ele não tem uma animação de aumentar de tamanho da matéria e fazer um sombreado aparecendo ao fundo para dar uma sensação de que ele se descolou da tela? Entao, ele utiliza exatamente essa propriedade CSS, só que muito provavelmente faz uso também da biblioteca bootstrap, com uma classe chamada CARD

Falei....falei e agora vou exemplificar:

```
.de2:hover{
 color: blue;
 background-color: red;
 width: 200px;
 height: 200px;
 border-left: 10px;
 border-left-style: dotted;
 border-color: black;
 padding-left: 10px;
 margin-left: 100px;
 box-sizing: border-box;
 margin-top: 50px;
 OVERFLOW: normal;
```

```
}
.de2{
  color: blue;
  background-color: red;
  width: 100px;
  height: 100px;
  border-left: 10px;
  border-left-style: dotted;
  border-color: black;
  padding-left: 10px;
  margin-left: 100px;
  box-sizing: border-box;
  margin-top: 50px;
  OVERFLOW: hidden
}
```

Ao usar propriedades na classe, se deve duplicar a classe, assim, por padrão a classe somente com .de2{} vai ser executada, agora quando o mouse passar pela DIV que faz uso da classe .de2, as propriedades do .de2:hover {} é que vão ser utilizadas.

Resultado:

=

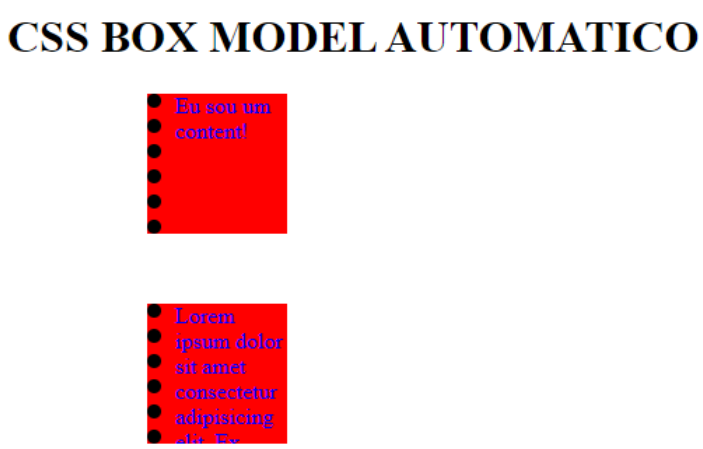


Figura 1 DIV sem interação do mouse

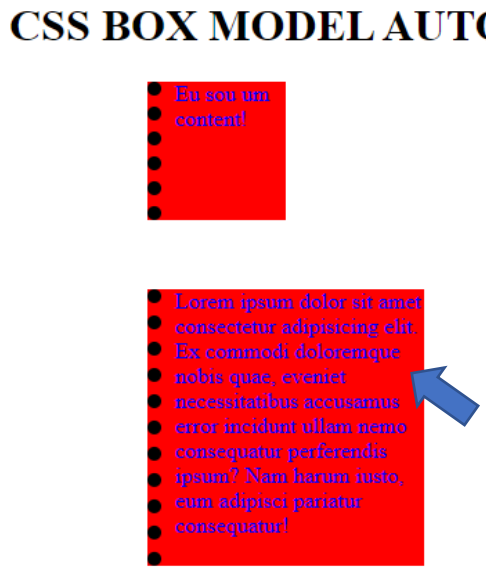


Figura 2 DIV com interação do mouse

Sites recomendados:

Bootstrap em Português · O mais popular framework front-end responsivo e focado para dispositivos móveis do mundo. (getbootstrap.com.br)

Guia de Unidades no CSS | Alura

W3Schools CSS overflow-wrap demonstration

CSS :hover Selector (w3schools.com)