2 - CSS Box Layout

linkedin.com/in/barbaraeste/
github.com/babieste/

Introdução ao modelo de formatação visual

Muitas das informações sobre o modelo de formatação visual são definidas no CSS2, no entanto, várias especificações de nível três (CSS3), como flex e grid layout foram adicionadas a estas informações. Ao ler as especificações, é frequente a referência ao modelo conforme o CSS2, portanto uma compreensão do modelo inicial é valiosa ao ler os outros modelos de layout.

O modelo de formatação visual descreve como os user agents, como por exemplo o browser, processam a árvore de documentos para mídias visuais. Cada elemento na árvore de documento gera zero ou mais boxes de acordo com o box model. O layout desses boxes é governado por:

1. Suas dimensões;
2. Seu tipo;
3. Esquema de posicionamento;
4. Relação entre os elementos na árvore de documentos;
5. Informações externas, como por exemplo o tamanho do viewport.

Containing blocks

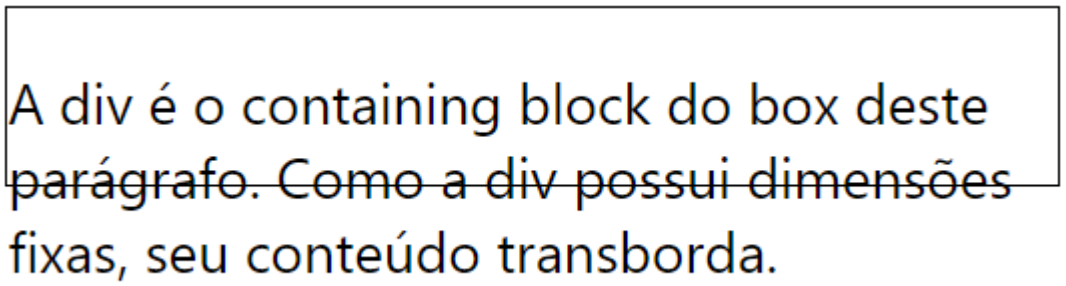
No CSS, posições e tamanhos de boxes são, em sua maioria, calculados em relação às bordas de um box denominado **containing block**. No geral, os boxes funcionam como blocos contendo boxes descendentes. Dizemos então que um box estabelece um containing block para seus descendentes. A frase **"o containing block de um box"** significa **"o bloco que contém o box"**, e não o bloco que aquele box gera.

O processo para identificar o containing block de um box depende totalmente do valor da propriedade '**position**' do elemento. Apesar de não entrarmos em detalhes sobre essa propriedade aqui, é importante entender:

1. O containing block do **root element** é um retângulo denominado **initial containing block**;
2. Se a 'position' do elemento é '**relative**' ou '**static**', o containing block é estabelecido pela content edge do ancestral mais próximo que é um block container ou que estabelece um contexto de formatação;
3. Se a 'position' do elemento é '**fixed**', o containing block é estabelecido pelo viewport;
4. Se a 'position' do elemento é '**absolute**', o containing block é estabelecido pelo ancestral mais próximo que possui 'position' de 'absolute', 'relative' ou 'fixed';
5. Se não há nenhum ancestral que siga estas regras, o containing block será o initial containing block.

Exemplo 1) Containing block com overflow:

```html
<div style="border: 1px solid black; height: 50px; width: 300px">
  <p>
    A div é o containing block do box deste parágrafo. Como a div
    possui dimensões fixas, seu conteúdo transborda.
  </p>
</div>
```



Exemplo 2) Containing blocks com elementos inline-level:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <title>Containing blocks</title>
  </head>
  <body>
    <div id="div-1">
      <p id="p-1">Lorem ipsum dolor sit amet</p>
      <p id="p-2">
        Fusce ultrices
        <em id="em-1">
          odio vel <strong id="strong-1">lorem bibendum</strong>
        </em>
      </p>
    </div>
  </body>
</html>
```

O documento acima estabelece os seguintes containing blocks:

Para o box gerado por	**O containing block é estabelecido por**
html	Containing block inicial
body	html
#div-1	body
#p-1	#div-1
#p-2	#div-1
#em-1	#p-2
#strong-1	#p-2

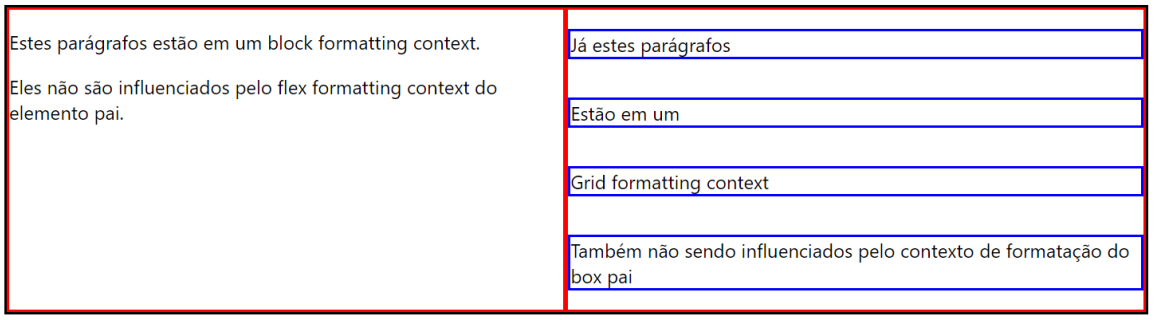
Formatting context

Um contexto de formatação é o ambiente ou área no qual um conjunto de boxes relacionados é disposto. Diferentes contextos de formatação organizam seus boxes de acordo com regras diferentes.

Por exemplo, um flex formatting context dispõe os boxes de acordo com as regras de flex layout, enquanto um block formatting context ou um inline formatting context dispõe os boxes de acordo com as regras de flow layout.

Um box pode estabelecer um novo contexto independente ou continuar o contexto do bloco que o contém. No entanto, salvo se especificado explicitamente, um contexto criado será sempre independente. Quando isso acontece, seja esse contexto do mesmo tipo que seu pai ou não, ele essencialmente cria um ambiente de layout novo: exceto através do dimensionamento do próprio box, o layout de seus descendentes não é afetado pelas regras e conteúdos do contexto de formatação fora do box e vice-versa.

```html
<div style="display: flex">
  <div style="display: block">
    <p>Estes parágrafos estão em um block formatting context.</p>
    <p>
      Eles não são influenciados pelo flex formatting context do
      elemento pai.
    </p>
  </div>
  <div style="display: grid">
    <p>Já estes parágrafos</p>
    <p>Estão em um</p>
    <p>Grid formatting context</p>
    <p>
      Também não sendo influenciados pelo contexto de formatação do box
      pai
    </p>
  </div>
</div>
```



Certas propriedades podem forçar um box a estabelecer um contexto de formatação independente em casos onde normalmente isso não aconteceria. Por exemplo, posicionar um box fora do fluxo com a propriedade 'position' ou transformar um bloco em um scroll container com a propriedade 'overflow' fará com que ele estabeleça um contexto de formatação independente.

Flow layout

O flow layout é a maneira como os elementos são exibidos em um viewport antes de qualquer alteração ser feita em seu layout. É em si um sistema de layout, assim como flex e o grid. Neste sistema, os elementos inline são exibidos na direção inline, ou seja, na direção em que as palavras são exibidas em uma frase de acordo com o modo de escrita do documento. Os elementos block são exibidos um após o outro, como fazem os parágrafos.

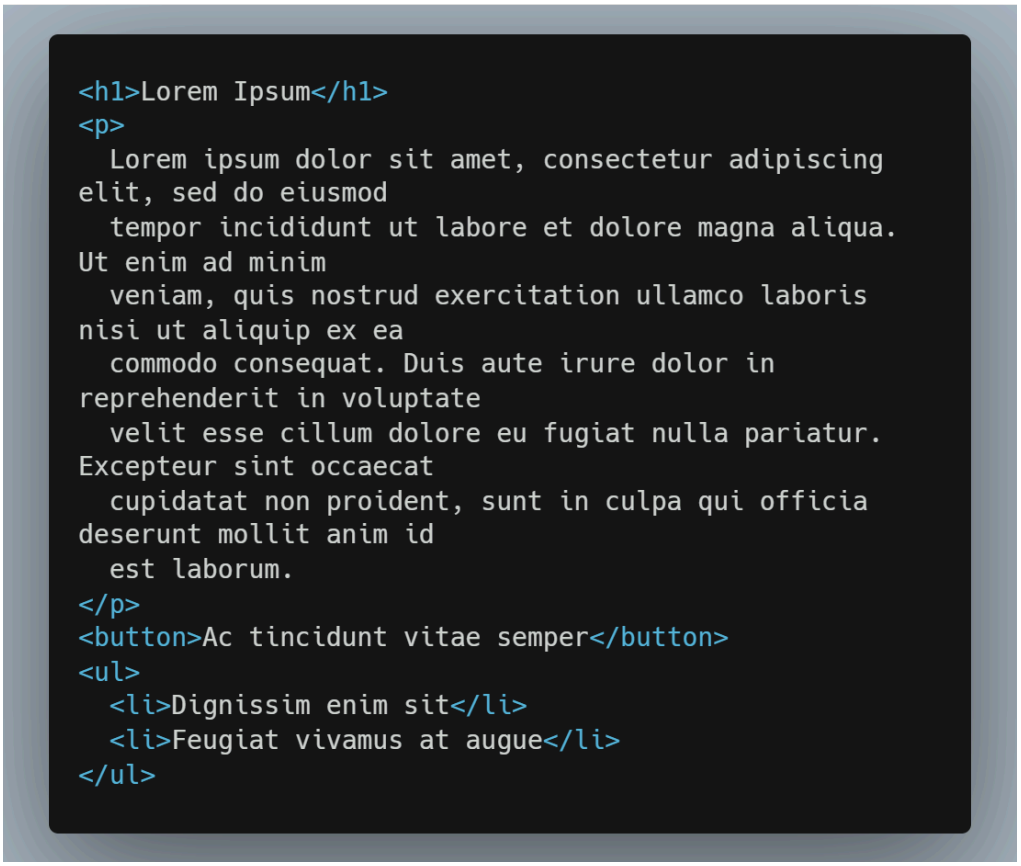
Os boxes no flow layout pertencem a um contexto de formatação, que pode ser block ou inline, mas nunca os dois ao mesmo tempo.

Block-level elements e block boxes

Block-level elements são elementos que geram um box principal que é block-level, isto é:

1. Sempre começam em uma nova linha;
2. Ocupam todo o espaço horizontal do containing block que estão inseridos;
3. Respeitam as propriedades de largura (width) e altura (height);
4. Margens horizontais e verticais funcionam normalmente.

Block-level boxes são os boxes gerados por block-level elements e participam em um block formatting context. Em um block formatting context, os boxes são dispostos verticalmente, um após o outro, no começo do containing block. A distância vertical entre dois boxes irmãos é definida pela margem.



```
<h1>Lorem Ipsum</h1>
<p>
  Lorem ipsum dolor sit amet, consectetur adipiscing
elit, sed do eiusmod
  tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim
  veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea
  commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate
  velit esse cillum dolore eu fugiat nulla pariatur.
Excepteur sint occaecat
  cupidatat non proident, sunt in culpa qui officia
deserunt mollit anim id
  est laborum.
</p>
<button>Ac tincidunt vitae semper</button>
<ul>
  <li>Dignissim enim sit</li>
  <li>Feugiat vivamus at augue</li>
</ul>
```

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Ac tincidunt vitae semper

- Dignissim enim sit
- Feugiat vivamus at augue

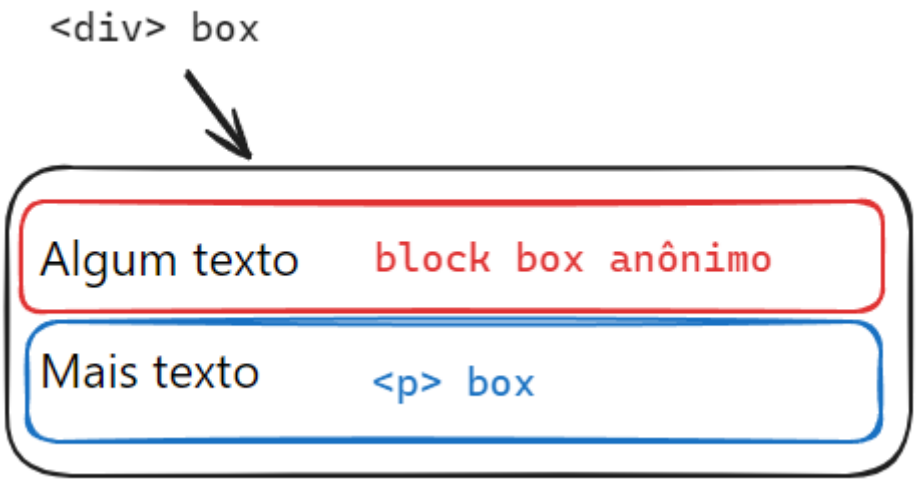
No block formatting context, os boxes são estabelecidos um após o outro.

Um block-level box também pode ser denominado **block container box** se contém apenas block-level elements, estabelecendo um block formatting context ou contém apenas inline-level elements, estabelecendo um inline formatting context. Block-level boxes que são block containers são denominados **block boxes**.

No documento abaixo, assumindo que ambas a DIV e o P possuem '**display: block**'. A DIV aparenta possuir ao mesmo tempo um conteúdo inline e um conteúdo block:

```
<div>
  Algum texto
  <p>Mais texto</p>
</div>
```

Todavia, se um block container box (como a DIV) possuir block-level boxes dentro (como o P), então força-se a ter somente block-level boxes. Para isso assume-se um block box anônimo que envolve "Algum texto", e o contexto de formatação será block:



Inline-level elements e inline boxes

Inline-level elements são os elementos que não formam blocos de conteúdo, ou seja, o conteúdo destes são distribuídos em linha. Suas características são:

1. Não começam em uma nova linha;
2. Utilizam somente o espaço horizontal necessário para seu conteúdo;
3. Não aceitam as propriedades ‘width’ e ‘height’;
4. Margens funcionam na dimensão inline, mas não na dimensão block;
5. Padding funciona em todos os lados, mas na dimensão inline podem sobrepor outros elementos.

Inline-level elements geram inline-level boxes, que são boxes que participam em um inline formatting context. Um inline formatting context é estabelecido por um block container box que não contém block-level boxes.

Um inline box é aquele que é inline-level e cujo conteúdo participa no inline formatting context no qual está inserido. Inline-level boxes que não são inline boxes são denominados inline-level boxes atômicos, porque eles participam em um inline formatting context como um único bloco (em outras palavras, não há quebra de linha em seu conteúdo).

Em um inline formatting context, os boxes são dispostos horizontalmente, um após o outro, no começo do containing block. Margens horizontais, bordas e paddings são respeitados entre os boxes. A área retangular que contém os boxes que formam uma linha é chamada **line box**.

A largura de uma line box é determinada pelo seu containing block. Uma line box é sempre alta o suficiente para todos os boxes que contém, no entanto, pode ser mais alta que o box mais alto que contém (se, por exemplo, os boxes estiverem alinhados

pelo seu baseline). Quando a altura de um box é menor que a altura da line box que o contém, seu alinhamento vertical é determinado pela propriedade '**vertical-align**'.

Quando diversos inline-level boxes não cabem horizontalmente em uma única line box, eles são distribuídos entre duas ou mais line boxes empilhadas verticalmente. Assim, um parágrafo é uma pilha vertical de line boxes. Se um inline box não puder ser dividido (por exemplo, se contiver um único caractere, ou se as regras de quebra de palavras específicas do idioma não permitirem uma quebra dentro do box), o box transbordará (overflow) a line box. Quando uma line box é quebrada, margens, paddings e bordas não possuem efeito visual na região onde a quebra ocorreu.

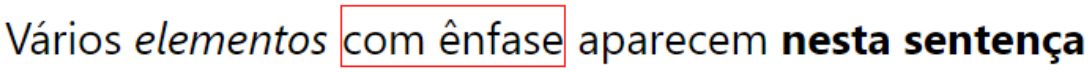
Exemplo:

```
<p>
  Vários <em>elementos</em>
  <span style="border: 1px solid red">com ênfase</span> aparecem
  <strong>nesta sentença</strong>
</p>
```

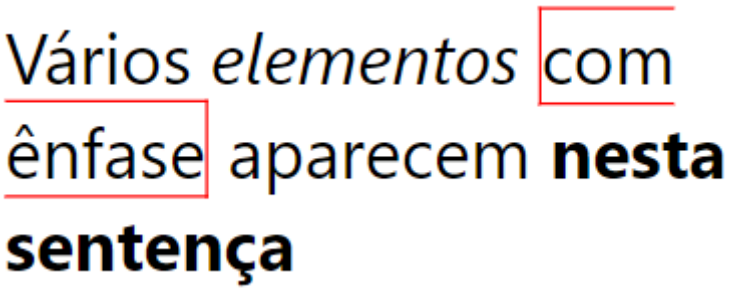
O elemento P gera um block box que contém cinco inline boxes:

1. Anônimo: "Vários";
2. : "elementos";
3. : "com ênfase";
4. Anônimo: "aparecem";
5. : "nesta sentença";

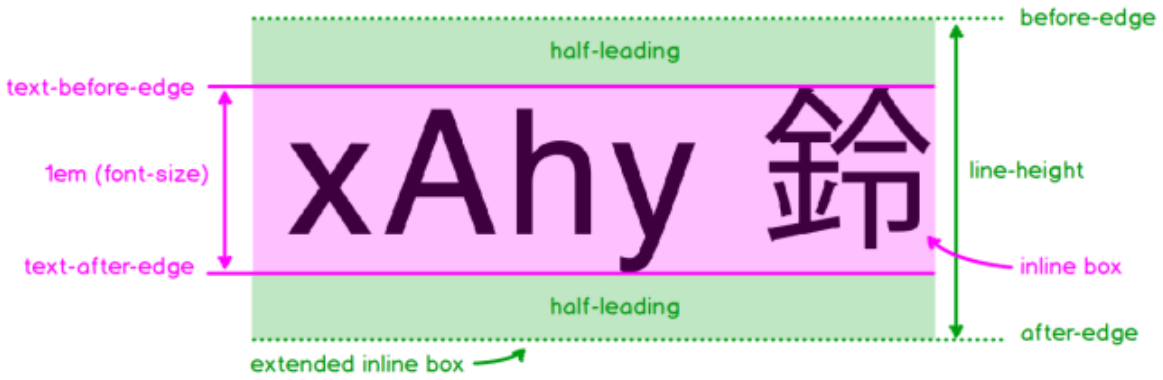
Para formatar o parágrafo, o browser coloca os elementos em line boxes, ou seja, o box gerado por P estabelece o containing block para as line boxes. Se o containing block é suficientemente espaçoso, todos os inline boxes encaixam em um único line box:



Caso contrário, os inline boxes irão se separarem em várias line boxes:



Quando a largura total dos inline-level boxes é menor do que a largura do line box que que os contém, sua distribuição horizontal é determinada pela propriedade '**text-align**'.



Fonte: https://html5experts.jp/takazudo/13339/

Calculando larguras no flow layout

A posição e o tamanho de um box é, na maioria das vezes, calculado relativo ao seu containing block. A propriedade '**width**' especifica a largura da **content area** dos boxes. Ela é aplicável a todos os tipos de elementos, com exceção de inline-level elements.

> A largura de um elemento inline-level é a do conteúdo renderizado dentro dele. Inline boxes são estabelecidos em line boxes, e a largura do line box é então definida pelo containing block.

Os valores para a propriedade podem ser:

1. Comprimento: especifica a largura da content area usando uma unidade de comprimento;
2. Porcentagem: a porcentagem é calculada em relação à largura do containing block que contém o box. Se a largura do containing block depende da largura deste elemento, então o layout resultante é indefinido.
3. 'auto': a largura depende do valor de outras propriedades definidas abaixo.

Os valores usados para as propriedades '**width**', '**margin-left**' e '**margin-right**' dependem do tipo de box gerado e entre as próprias propriedades:

1. Para inline-level elements, a propriedade 'width' não é aplicável;
2. Para block-level elements, a seguinte restrição deve ser válida:

```
margin-left + border-left-width + padding-left + width + padding-right + border-right-width + margin-right
=
width do containing block
```

a. Se ‘width’ não é ‘auto’ e a soma ‘border-left-width’ + ‘padding-left’ + ‘width’ + ‘padding-right’ + ‘border-right-width’ é maior do que a largura do containing block, então qualquer valor computado de ‘auto’ aplicado para ‘margin-left’ ou ‘margin-right’ será zero;

b. Se todas as propriedades acima possuem um valor computado diferente de ‘auto’ e a sua soma for maior do que a largura do containing block, então um dos valores utilizados será diferente do valor computado.

    i. Se a ‘direction’ do containing block for ‘ltr’, então o valor especificado para ‘margin-right’ é ignorado e o valor usado é calculado de forma a fazer a equalidade verdadeira. Se o valor de ‘direction’ é ‘rtl’, então isso acontece com ‘margin-left’;

c. Se uma das propriedades ‘margin-left’ ou ‘margin-right’ é ‘auto’, então o valor usado é resultante da equação;

d. Se ‘width’ é ‘auto’, então qualquer outro valor computado de ‘auto’ se torna zero e o valor usado para ‘width’ é resultante da equação;

e. Se ambas ‘margin-left’ e ‘margin-right’ são ‘auto’, então o valor usado para ambas é igual, o que centraliza o elemento com respeito ao seu containing block.

Larguras máximas e mínimas

As propriedades ‘**min-width**’ e ‘**max-width**’ permitem restringir a largura do conteúdo a um certo intervalo. Valores negativos não são permitidos. Os valores podem ser:

1. Comprimento: especifica um valor usado máximo ou mínimo fixo para a largura;
2. Porcentagem: especifica uma porcentagem para o valor usado. A porcentagem é calculada com respeito à largura do containing block. Se a largura do containing block é negativa, o valor usado é zero;
3. ‘none’: Não há limite (superior) para a largura do box.

O seguinte algoritmo descreve como as duas propriedades influenciam o valor usado para a propriedade ‘width’:

1. O valor definido para ‘width’ é tentado, aplicando as regras descritas na seção anterior;
2. Se o valor é maior do que ‘max-width’, as regras são aplicadas novamente, mas desta vez usando o valor de ‘max-width’;
3. Se o valor é menor do que ‘min-width’, as regras são aplicadas novamente, mas desta vez usando o valor de ‘min-width’;

Calculando alturas no flow layout

A propriedade '**height**' especifica a altura da **content area** de um box. Valores negativos para altura não são permitidos. Esta propriedade não se aplica para inline elements. Os valores podem ser:

1. Comprimento: especifica a altura da content area usando uma medida de comprimento;
2. Porcentagem: a porcentagem é calculada com respeito à altura do containing block. Se a altura do containing block não é especificada explicitamente (ou seja, depende do conteúdo), então o valor usado é calculado como se fosse especificado 'auto';
3. 'auto': a altura depende de valores de outras propriedades, definidas abaixo.

Para calcular os valores de '**height**', '**margin-top**' e '**margin-bottom**', é necessário distinguir entre os tipos de boxes:

1. Para inline-level elements, a propriedade 'height' não se aplica. A altura da content area nesses casos deve basear-se na fonte utilizada, mas a especificação de CSS não determina como isso deve ser feito (ou seja, cabe ao user agent definir esse cálculo);
2. Para block-level elements:
    a. Se 'overflow' é 'visible':
        i. Se 'margin-top' ou 'margin-bottom' é 'auto', então o valor usado é zero;
        ii. Se 'height' é 'auto', então a altura depende se o elemento possui algum elemento block-level e se este possui padding ou bordas.
    b. Se 'overflow' for diferente de 'visible':
        i. Se 'margin-top' ou 'margin-bottom' é 'auto', então o valor usado é zero;
        ii. Se 'height' é 'auto':
            1. Se o elemento possui apenas descendentes inline-level, então a altura será a distância da parte superior da line box mais alta à parte inferior da line box mais baixa;
            2. Se possui descendentes block-level, então a altura será a distância entre a borda da margem superior do descendente block-level mais alto e a borda da margem inferior do descendente block-level mais baixo.

Alturas máximas e mínimas

As propriedades ‘**min-height**’ e ‘**max-height**’ permitem restringir a altura de um box a um certo intervalo. Valores negativos não são permitidos. Os valores podem ser:

1. Comprimento: especifica um valor máximo ou mínimo fixo para a altura;
2. Porcentagem: a porcentagem é calculada com relação à altura do containing block. Se a altura do containing block não é especificada explicitamente (ou seja, depende do conteúdo), então o valor é tratado como zero, para ‘min-height’, ou ‘none’, para ‘max-height’;
3. ‘none’: não há limite (superior) para a altura do box.

O seguinte algoritmo descreve como as duas propriedades influenciam o valor usado para a propriedade ‘height’:

1. O valor definido em ‘height’ é utilizado, seguindo as regras descritas na seção anterior;
    a. Se o valor é maior do que ‘max-height’, então as regras são aplicadas novamente, mas agora com ‘max-height’;
    b. Se o valor é menor do que ‘min-height’, então as regras são aplicadas novamente, mas agora com ‘min-height’.

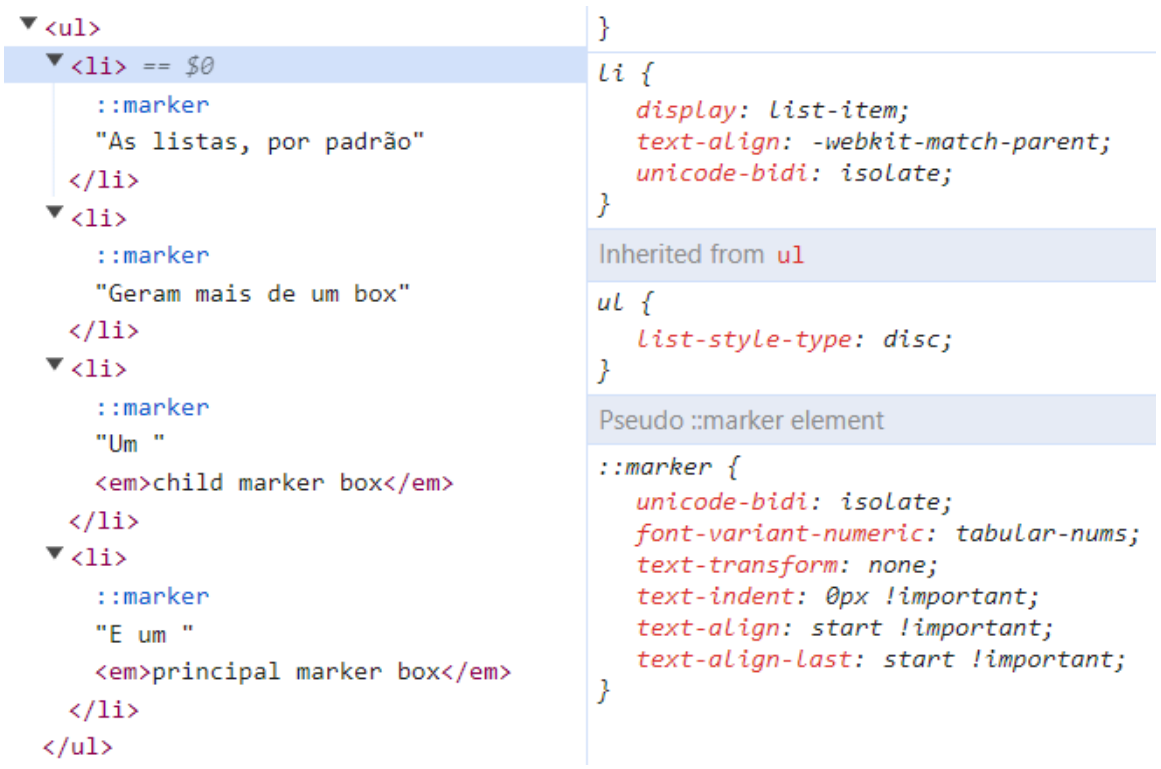
Box Layout Modes: a propriedade display

Controlando a geração de boxes

A geração de boxes é a parte do modelo de formatação visual que cria boxes a partir de elementos. O tipo de um box afeta, em parte, seu comportamento no modelo de formatação visual. A propriedade ‘**display**’ especifica o tipo de um box.

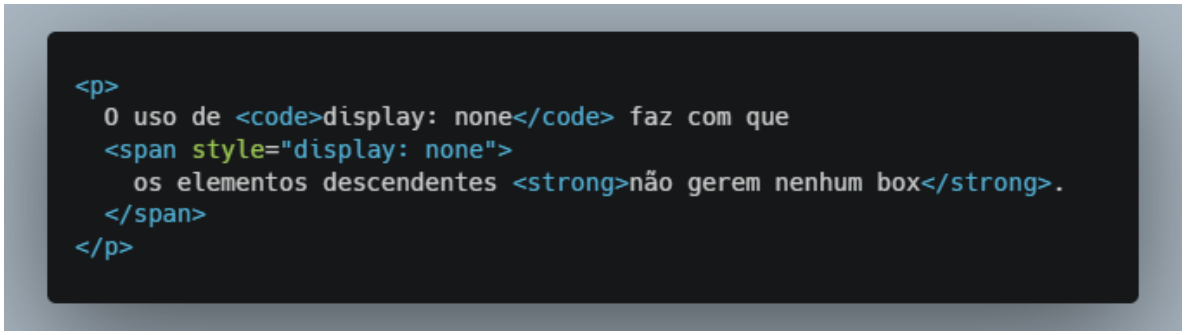
Como sabemos, o CSS gera uma estrutura intermediária, a box tree, que representa a formatação do documento renderizado. Cada box representa seu elemento correspondente no documento. Então, para cada elemento, o CSS gera zero ou mais boxes, conforme especificado pelo display.

Geralmente um elemento gera um único box, o **box principal**, que representa o próprio elemento e contém os boxes descendentes. Mas alguns tipos de ‘display’ podem gerar boxes adicionais, por exemplo itens de listas, que possuem por padrão o ‘display’ com valor ‘list-item’:



- As listas, por padrão
- Geram mais de um box
- Um *child marker box*
- E um *principal marker box*

E alguns valores, como ‘display: none’, causam o elemento a não gerar nenhum box:



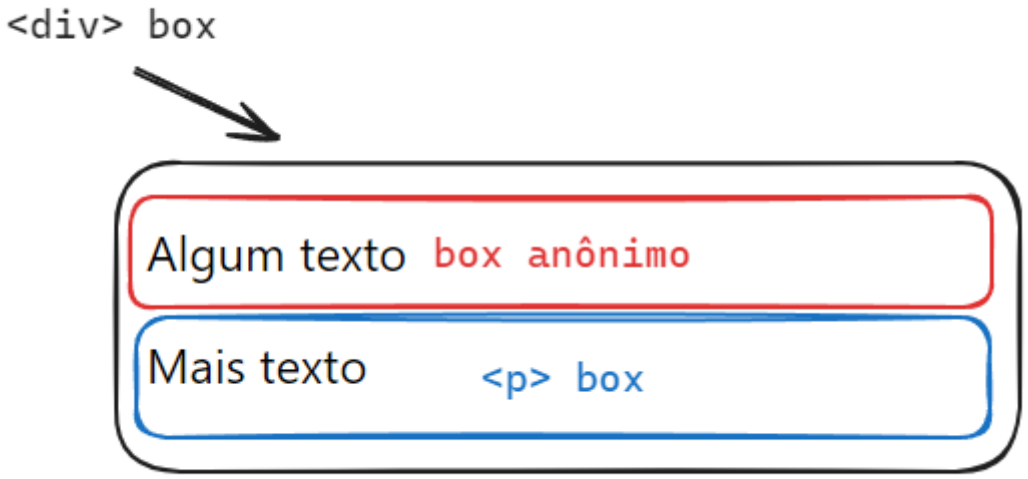
O uso de `display: none` faz com que

Na construção da box tree, boxes gerados por um elemento são descendentes do principal box de algum elemento ancestral. No caso geral, o box pai direto do

principal box de um elemento é o principal box do ancestral mais próximo que gera um box.

Um **box anônimo** é aquele que não é associado com nenhum elemento. São gerados em certas circunstâncias para ajustar a box tree quando ela requer uma estrutura que os boxes gerados pela árvore de elementos não proverá.

```
<div>
  Algum texto
  <p>Mais texto</p>
</div>
```



Exemplo de box anônimo

A propriedade ‘display’ define o tipo do box, que consiste:

1. No **outer display type**, definindo como o principal box em si participa no flow layout;
2. No **inner display type**, definindo o tipo contexto de formatação que o box gera, ditando como os boxes descendentes são dispostos.

Alguns valores para display		
Short display	**Full display**	**Tipo de box gerado**
‘none’	‘none’	Sub-árvore é omitida da box tree

‘contents’	‘contents’	O elemento é substituído pelo seu conteúdo na box tree
‘block’	‘block flow’	Block-level block container
‘inline’	‘inline flow’	Inline-level box
‘inline-block’	‘inline flow-root’	Inline-level block container
‘flex’	‘block flex’	Block-level flex container
‘inline-flex’	‘inline flex’	Inline-level flex container
‘grid’	‘block grid’	Block-level grid container
‘inline-grid’	‘inline grid’	Inline-level grid container

Outer display

O **outer display type** de um elemento define o papel do seu principal box no flow layout. São estes:

1. ‘**block**’: o elemento gera um box que é block-level;
2. ‘**inline**’: o elemento gera um box que é inline-level.

Se um valor para o outer display type é especificado mas o valor para o inner display type é omitido, então o inner display type do elemento será ‘flow’.

Inner display

O **inner display type** de um elemento especifica o tipo de contexto de formatação ao qual seu conteúdo será submetido. São alguns destes:

1. ‘**flow**’: O elemento apresenta seu conteúdo usando flow layout. Se o outer display type é inline, e está participando de um block ou inline formatting context, então gera um inline box; Caso contrário, gera um block container box.
2. ‘**flow-root**’: Parecido com o flow, mas sempre estabelece um novo block formatting context para seu conteúdo;
3. ‘**flex**’: O elemento gera um flex container box e estabelece um flex formatting context;

4. ‘**grid**’: O elemento gera um grid container box e estabelece um grid formatting context;

Se um valor para o inner display type é especificado mas o valor para o outer display type é omitido, então o outer display type do elemento será block.

Referências

https://www.w3.org/TR/CSS22/visuren.html#visual-model-intro
https://www.w3.org/TR/CSS22/visudet.html
https://developer.mozilla.org/en-US/docs/Web/CSS/Visual_formatting_model
https://www.w3.org/TR/css-display-3/#intro
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flow_layout