Ministério da Educação
Universidade Federal de Itajubá – *Itajubá*
COM222/XDESC03 – Programação Web/Sistemas Web
Aula – 6 – Modelo de Caixa
Prof.: Phyllipe Lima

UNIFEI

Objetivos:

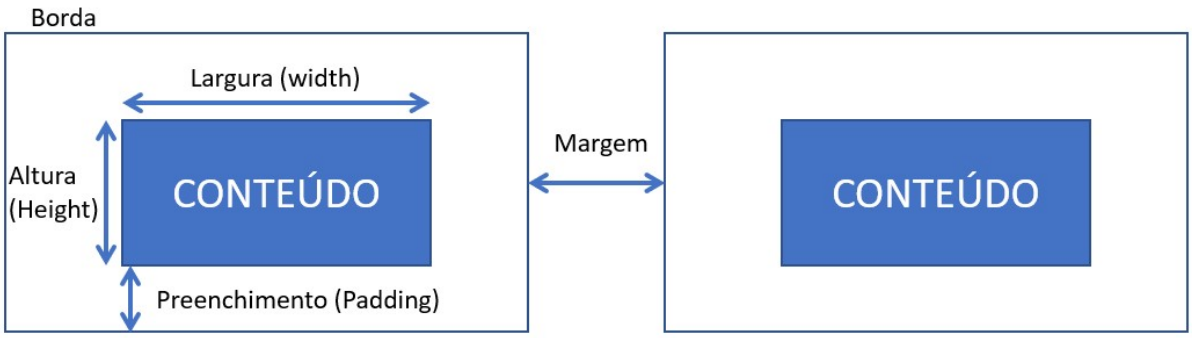
1. Compreender o Modelo de Caixa e as propriedades

Conteúdo Programático

1. Modelo de Caixa com as propriedades
2. Unidades de Medidas Absolutas e Relativas
3. Display em linha e bloco
4. Posicionamento
5. Exemplos práticos

1 -) O que é o Modelo de Caixa? (Box Model)

O modelo de caixa é o conceito básico usado pelo CSS para representar elementos HTML na tela. Cada elemento HTML é considerado como uma caixa retangular que contém conteúdo, preenchimento (*padding*), borda (*border*) e margem (*margin*). Essas quatro componentes são calculadas e renderizadas pelo navegador. Todos os esses componentes possuem valores padrões para cada elementos, mas conseguimos modificar explicitamente usando regras de estilização. A Figura abaixo apresenta o modelo de caixa.



O conteúdo é a parte interna da caixa retangular, que contém o texto, imagem, botão ou outros elementos HTML que desejamos apresentar na tela. O preenchimento é a área entre o conteúdo e a borda, que pode ser usado para adicionar espaço ao redor do conteúdo. A borda é a linha mais externa da caixa retangular que contém o conteúdo e pode ter um estilo, largura e cor definidos pelo CSS (propriedade *border*). A margem é a área externa da borda, que é usada para separar elementos uns dos outros.

Para entendermos melhor o que é o modelo de caixa vamos criar um elemento qualquer e usar o navegador para fazer a inspeção. Considere o código HTML abaixo.

```
<body>
    <p>Modelo de Caixa</p>
</body>
```

Ao inspecionarmos esse elemento no navegador, apenas pairando (*hover*) sobre o próprio temos a Figura 1 (OBS: imagem gerada com inspeção do Microsoft Edge).

Figura 1

Usando a inspeção, podemos ver os valores de margem, borda e preenchimento. A Figura 2 apresenta as medidas do elemento <p> sendo analisado. Veja que a borda e preenchimento estão com - (significa automática, nesse exemplo 0 - zero), pois não definimos valores e o elemento <p>, por padrão, possui esses valores zerados. Apenas a margem que possui um valor de 16px, para garantir que o elemento esteja em bloco. Isso significa que esse elemento <p> estará 16px de distância de outro elemento, mas isso a partir de sua borda e não a partir do conteúdo. O conteúdo está em um retângulo com largura 366px e altura 18px. Esses valores do retângulo do conteúdo se modificam conforme o próprio conteúdo a ser exibido se altera, além de redimensionamentos no navegador.

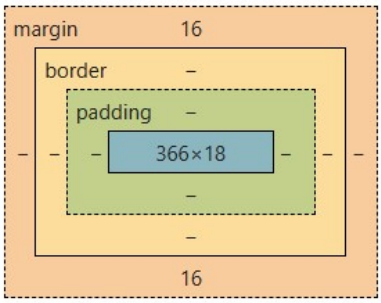


Figura 2

1.1 – Largura e Altura

Podemos definir a altura (height) e largura (width) de um elemento atribuindo valores diretamente nessas propriedades. Essas medidas irão definir a área retangular com o conteúdo. É o retângulo mais interno. Normalmente aparece em azul quando estamos observando pelo inspetor dos navegadores.

Exemplo, para definir o elemento <p> com largura e altura de 200px fazemos:

```
p{
   width: 200px;
   height: 200px;
}

<p>
   Lorem, ipsum dolor sit amet consectetur adipisicing elit. Illum deserunt quam repellendus debitis nisi veniam minus quo,
   eligendi corrupti beatae libero voluptatum sit officia fugiat   numquam nostrum totam! Dolorum, vel.
</p>
```

A Figura 3 apresenta o resultado no navegador com o inspetor. Observe que o retângulo mais interno está azul e o conteúdo do texto não preenche totalmente a altura. Mesmo assim, o espaço de 200px foi reservado pois definimos no CSS os valores.

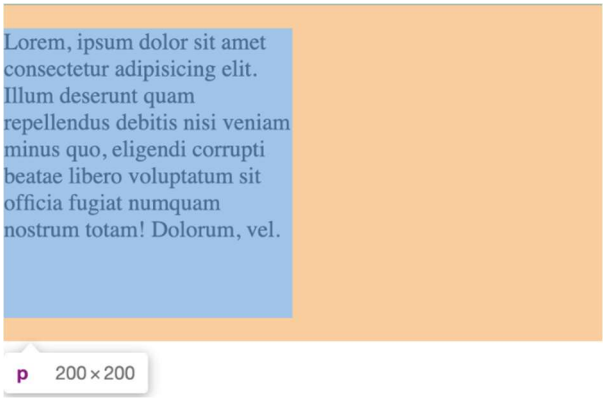


Figura 3

1.2 – Preenchimento (padding)

O preenchimento é valor que será adicionado entre o conteúdo e a borda, isto é, o preenchimento fica dentro do elemento HTML. Podemos modificar o preenchimento usando as seguintes propriedades.

```
/* Modifica o preenchimento superior */
padding-top: ;
/* Modifica o preenchimento a direita */
padding-right: ;
/* Modifica o preenchimento inferior */
padding-bottom: ;
/* Modifica o preenchimento a esquerda */
padding-left: ;
/* É capaz de modificar todo o preenchimento.
Se colocado um valor, este será aplicado nos 4 lados.
Se colocado dois valores, o primeiro é aplicado verticalmente (top e bottom)
e o segundo horizontalmente (left e right)
      Se colocado 4 valores, estes serão atribuidos de acordo com a ordem top right
bottom left
*/
padding:
```

Modifique os valores das propriedades acima e veja o resultado no navegador. Utiliza a inspeção.

1.3 – Margem

A margem representa a distância externa a borda. É ela quem irá determinar qual distante o elemento ficará dos demais. Para controlar a margem usamos a propriedade “margin”. Ela segue o mesmo padrão de atribuição de valores que o preenchimento (padding).

1.4 – Borda

A borda representa o retângulo mais externo do elemento HTML. É a borda quem define o “fim” do elemento. A princípio a borda fica invisível e precisa ser modificada para que se torne presente/visível. Ela é usada para destacar ou enfatizar elementos na página e pode ser estilizada de várias maneiras, incluindo largura, cor e estilo.

Para adicionar uma borda a um elemento HTML, é preciso usar a propriedade *border*. Assim como *padding,* a *border* possui várias formas de ser quebrada. No caso da borda é necessário adicionar uma largura, estilo e cor.

```
/* controla a largura da borda. Pode ser definida como um valor específico em px, uma palavra-chave como "thin" ou "thick", ou uma combinação usando 4 valores na ordem top right bottom left */
border-width: ;

/* controla o estilo da borda. Pode ser definida como uma palavra-chave como "solid", "dotted" ou "dashed", ou como uma combinação de palavras-chave usando a notação top right bottom left. */
border-style: ;

/* controla a cor da borda. Pode ser definida como um valor de cor específico ou usandos as fórmulas como RGB, Hexadecimal e outros.. */
border-color: ;

/* As três propriedades acima podem ser aplicadas simultaneamente utilizando apenas "border" */
border : ;
```

Importante notar que se for utilizar apenas a propriedade “border”, é necessário passar os valores de forma coerente e seguindo, preferencialmente a ordem: largura, estilo e cor. Em largura é necessário usar um valor para definir as 4 larguras.

Exemplo 1: Criar uma borda sólida, azul com largura 5px

```
border: 5px solid rgb(0,0,255);
```

A Figura 4 apresenta o resultado aplicando essa estilização no elemento <p>, já combinados com *width* e *height* de 200px.

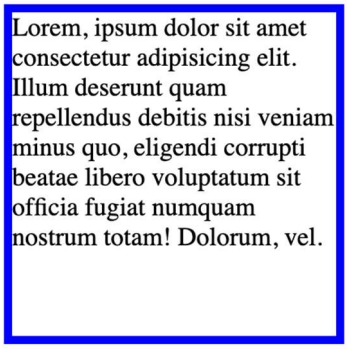


Figura 4

1.5 – Borda Arredondada

Para criar o efeito de borda arredondada podemos manipular o raio da borda, através da propriedade “border-radius”. Ela pode ser valores relativos (como %) como absolutos (px,pt). Se o elemento possuir a mesma largura e altura, aplicando o “border-radius” para 50%, teremos um efeito de um círculo. Importante observar que é necessário ter uma borda antes de aplicar o border-radius.

Exemplo 2: Criar uma borda arredondada com 10px

```
border-radius: 10px;
```

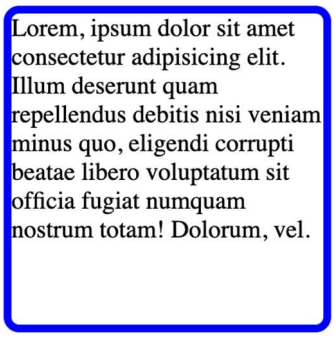


Figura 5

Exemplo 3: Criar uma borda tracejada, vermelha que se apresente como um círculo.

```
border: 5px dashed rgb(255,0,0);
border-radius: 50%;
```

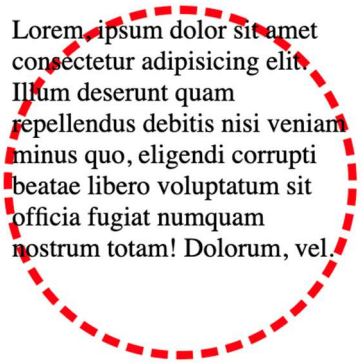


Figura 6

Importante destacar que o texto não está totalmente contido, pois o conteúdo é retangular e apenas criamos uma borda circular, que na realidade não representa a margem. Existem técnicas para resolvermos isso.

Exemplo 4: Escreva um código HTML/CSS para reproduzir o conteúdo na Figura 7.
Dica: Brinque com valores de preenchimento e margem.

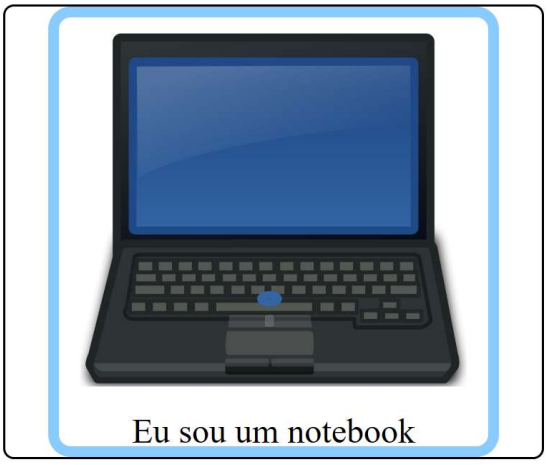


Figura 7

2 -) Medidas

2.1 – Medidas Absolutas

PX (CSS pixel): Unidade absoluta mais utilizada. A confusão feita é que 1px não é necessariamente 1 pixel no monitor, definido como a menor unidade endereçável. Não é recomendado para sites responsivos. Observando a definição do MDN:

"O Pixel CSS - denotado no CSS pelo sufixo px - é uma unidade de comprimento que corresponde aproximadamente a largura ou altura de um ponto único que pode ser confortavelmente visto pelos olhos humanos sem esforço, mas é o menor possível. Por definição, esse é o tamanho físico de um único píxel em uma densidade de 96 DPI, situado a um braço de distância dos olhos do observador.

Essa definição, é claro, se demonstra ser muito vaga, e termos como "confortavelmente visto" e "braço de distância" são imprecisas, variando de pessoa para pessoa. Por exemplo, quando um usuário se senta em sua mesa, defronte a sua área de trabalho, o monitor geralmente estará longe dos seus olhos enquanto ele estiver usando o celular."

Ou seja, a definição é vaga e basta entendermos que é um valor absoluto e não deverá ser usado exaustivamente e, provavelmente, aplicações reais darão preferência para unidades relativas.

2.2 – Medidas Relativas

% (porcentagem): Unidade relativa que ocupa o espaço em relação a uma porcentagem do elemento "pai".

EM: Unidade relativa ao *tamanho da fonte* aplicado no elemento "pai". Se o elemento pai possui uma fonte de 16px, e o filho uma fonte de "1em", este será 16px. Mas se for "2em" este será 32px. Conforme os objetos são aninhados, o valor de EM pode se alterar, pois ele sempre é calculado em relação ao pai. De modo que é possível os valores podem crescer rapidamente conforme se aninham elementos baseado na unidade EM.

REM: Unidade relativa ao *tamanho da fonte* aplicado no elemento "raiz". Se o elemento raiz possui uma fonte de 16px, e o filho uma fonte de "1rem", este será 16px. Mas se for "2rem" este será 32px. Conforme os objetos são aninhados, o valor de REM não se altera, pois ele sempre é calculado em relação ao elemento raiz.

Exemplo 5: O código abaixo apresenta uma comparação entre PX e %. Observe que os valores em PX, permanecem os mesmos em ambas as situações. Mas o valor % se modifica de acordo com o pai, pois a porcentagem é calculada sempre em relação ao elemento pai.

Código HTML:

```
<div class="caixa cinq-pr">50%</div>
<div class="caixa cinq-px">50px</div>
<hr>
<div class="pai">
   <div class="caixa cinq-pr">50%</div>
   <div class="caixa cinq-px">50px</div>
</div>
```

Código CSS:

```
.caixa{
    border: 1px solid black;
}
```

```css
.cinq-pr{
   width: 50%;
}

.cinq-px{
    width: 50px;
}

.pai {
    width: 100px;
    border: 1px solid black;
    padding-right: 15px;
}
```

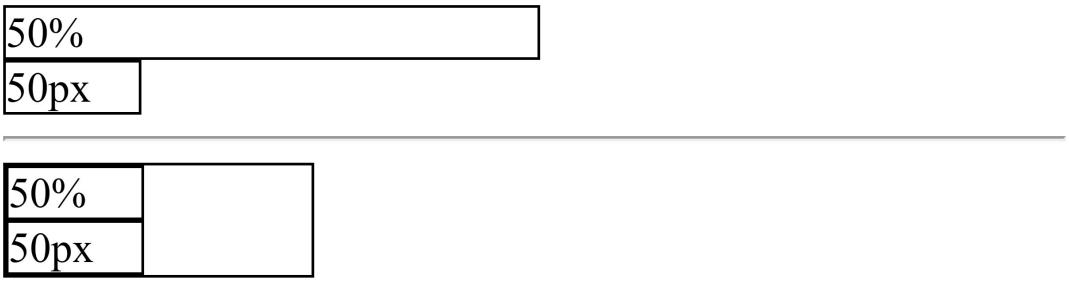


Figura 8

Exemplo 6: O código HTML/CSS abaixo apresenta a diferença entre EM e REM utilizando o exemplo de tamanho de fontes:

HTML:

```html
<body>
    <p class="um-em">Eu sou 1EM</p>
    <p class="um-rem">Eu sou 1REM</p>
    <p class="dois-em">Eu sou 2EM</p>
    <p class="dois-rem">Eu sou 2REM</p>

    <hr>

    <div class="caixa">
        <p class="um-em">Eu sou 1EM</p>
        <p class="um-rem">Eu sou 1REM</p>
        <p class="dois-em">Eu sou 2EM</p>
        <p class="dois-rem">Eu sou 2REM</p>
    </div>
</body>
```

CSS:

```css
.um-em{
    font-size: 1em;
}

.um-rem{
    font-size: 1rem;
```

```
}


.dois-em{
    font-size: 2em;
}

.dois-rem{
    font-size: 2rem;
}

.caixa{
    border: 1px solid black;
    font-size: 2em;
}
```

Figura 9

3 -) Display

A propriedade *display* no CSS é usada para controlar a forma que um elemento HTML é exibido na página. Ela pode ser usada para mudar o comportamento padrão de um elemento e torná-lo mais flexível de acordo com às necessidades. Esses são os valores que essa propriedade pode assumir.

```
display: block ;
 /* O elemento é exibido como um bloco, ocupando toda a largura disponível e adicionando uma quebra de linha antes e depois do elemento. O valor "block" é padrão para algins elementos como: div, p, h1, h2, etc. */

display: inline;
 /* O elemento é exibido como uma linha, permitindo que outros elementos fiquem lado a lado. Não adiciona quebra de linha antes ou depois do elemento. O valor "inline" é padrão para  elementos de texto, como span, a, em, strong, etc.
Quando o valor é "inline" as propriedades width e height são ignoradas */

display: inline-block;
 /* O elemento é exibido como uma linha, mas com a capacidade de ter largura e altura definidas. Isso permite que o elemento tenha conteúdo dentro dele e também possa ser estilizado. É geralmente usado para botões, imagens, listas de itens, etc. */

display: none; /* O elemento não é exibido na página. Ele é completamente removido do fluxo de layout e não ocupa espaço. */

display: flex; /*Faz o elemento ser exibido como um contêiner flexível. Será melhor explicado quando falarmos  do "flexbox*/
```

Exemplo 7: Crie um código para exibir 3 (três) quadrados lado a lado conforme Figura 10.

Figura 10

4 -) Posicionamento

A propriedade “position” permite definir como um elemento deve ser posicionado na página em relação aos outros elementos. A propriedade pode assumir os seguintes valores: static, relative, absolute, fixed e sticky.

```
/*Padrão para todos os elementos. Segue o fluxo normal do HTML e não é afetado pelas propriedades top, right, bottom e left. Normalmente não é definida explicitamente  */
position: static;

/* O elemento se posiciona em relação à sua posição original na página. É afetado pelas propriedades top, right, bottom e left. */
position: relative;

/* O elemento é removido do fluxo normal da página e é posicionado em relação ao seu elemento pai mais próximo que tenha a propriedade position definida explicitamente (menos static). É afetado pelas propriedades top, right, bottom e left para definir a posição do elemento em relação a esse elemento pai, "ancestral mais próximo" */
position: absolute;

/* O elemento é removido do fluxo normal da página e é posicionado em relação à janela do navegador. Ele permanece na mesma posição, mesmo que a página seja rolada. É afetado pelas propriedades top, right, bottom e left. */
position: fixed;

/* Posição que combina as características de relative e fixed. Quando definimos um elemento como "sticky", ele se comporta como relative até que alcance uma posição de referência, onde se torna fixed. Essa posição de referência pode ser definida utilizando a propriedade top, right, bottom ou left. */
position: sticky;
```

Exemplo 8: Crie um código HTML/CSS para criar uma barra de navegação que fique fixa no “header” da página. Use como exemplo a Figura 11.

Lorem ipsum dolor sit amet consectetur adipisicing elit. Sint ipsum officia nobis necessitatibus, eligendi inventore ad tempore animi placeat delectus id voluptates beatae

Figura 11