1º ESPAÇAMENTOS E DIMENSÕES

Temos algumas maneiras de trabalhar com dimensões e espaçamentos. Para espaçamento interno e externo usamos respectivamente padding e margin , e para redimensionar elementos podemos usar as propriedades de largura e altura ou width e height . Vamos ver mais a fundo essas propriedades.

DIMENSÕES

É possível determinar as dimensões de um elemento, por exemplo:

```
p {background-color: red;
height: 300px;
width: 300px;}
```

Todos os parágrafos do nosso HTML ocuparão 300 pixels de altura e de largura, com cor de fundo vermelha.

ESPAÇAMENTOS

Padding

A propriedade **padding** é utilizada para definir um espaçamento interno em elementos (por espaçamento interno queremos dizer a distância entre o limite do elemento, sua borda, e seu respectivo conteúdo) e tem as subpropriedades listadas a seguir:

- padding-top
- padding-right
- padding-bottom
- padding-left

Essas propriedades aplicam uma distância entre o limite do elemento e seu conteúdo acima, à direita, abaixo e à esquerda respectivamente. Essa ordem é importante para entendermos como funciona a *shorthand property* (encurtamento) do padding.
Podemos definir todos os valores para as subpropriedades do padding em uma única propriedade, chamada exatamente de `padding` , e seu comportamento é descrito nos exemplos a seguir:
Se passado somente um valor para a propriedade `padding` , esse mesmo valor é aplicado em todas as direções.

```
p {
padding: 10px;
}
```

Se passados dois valores, o primeiro será aplicado acima e abaixo (equivalente a passar o mesmo valor para `padding-top` e `padding-bottom`) e o segundo será aplicado à direita e à esquerda (equivalente ao mesmo valor para `padding-right` e `padding-left`).

```
p {
padding: 10px 15px;
}
```

Se passados três valores, o primeiro será aplicado acima (equivalente a padding-top), o segundo será aplicado à direita e à esquerda (equivalente a passar o mesmo valor para padding-right e padding-left) e o terceiro valor será aplicado abaixo do elemento (equivalente a padding-bottom).

```
p {
padding: 10px 20px 15px;
}
```

Se passados quatro valores, serão aplicados respectivamente a padding-top , padding-right , padding-bottom e padding-left . Para facilitar a memorização dessa ordem, basta lembrar que os valores são aplicados em **sentido horário**.

```
p {
padding: 10px 20px 15px 5px;
}
```

Uma dica para omitir valores do padding:

Quando precisar omitir valores, sempre omita no sentido anti-horário começando a partir da sub propriedade -left .

Como os valores têm posicionamento fixo na hora de declarar os espaçamentos, o navegador não sabe quando e qual valor deve ser omitido. Por exemplo:

Se tivermos um padding:

```
h1 {
padding: 10px 25px 10px 15px;
}
```

O código não pode sofrer o encurtamento porque por mais que os valores de top e bottom

sejam iguais, os valores right e left não são e eles são os primeiros a serem omitidos. Veja o

que acontece quando vamos omitir o valor de 10px do bottom:

```
h1 {
padding: 10px 25px 15px;
}
```

O navegador vai interpretar da seguinte maneira:

```
h1 {
padding: top right bottom;
}
```

Que no final vai ficar igual a:

```
h1 {
padding-top: 10px;
padding-right: 25px;
padding-bottom: 15px;
padding-left: 25px;
}
```

E esses valores não são os que nós colocamos no começo com padding: 10px 25px 10px

15px;

Margin

A propriedade margin é bem parecida com a propriedade padding , exceto que ela adiciona espaço após o limite do elemento, ou seja, é um espaçamento além do elemento em si (espaçamento externo). Além das subpropriedades listadas a seguir, há a *shorthand property* margin que se comporta da mesma maneira que a *shorthand property* do padding vista no tópico anterior.

- margin-top
- margin-right
- margin-bottom
- margin-left

Há ainda uma maneira de permitir que o navegador defina qual será a dimensão da propriedade
padding ou margin conforme o espaço disponível na tela: definimos o valor auto para os espaçamentos que quisermos.
No exemplo a seguir, definimos que um elemento não tem nenhuma margem acima ou abaixo de seu conteúdo e que o navegador define uma margem igual para ambos os lados de acordo com o espaço disponível:

```
p {
margin: 0 auto;
}
```

OBJETIVO 1:

- diminuir a largura do conteúdo da página para **940px**
- Para que o conteúdo não fique colado no canto da página, afaste ele dos cantos direito e esquerdo
- da tela, até que fique centralizado.
- criar um espaço de respiro interno de **10px** para o conteúdo do título principal
- criar um espaço de respiro interno de **15px** para o conteúdo das figuras
- afastar **30px** as figuras dos cantos da tela e das caixas que vêm antes e depois delas
- dentro das figuras, afastar **10px** a legenda da imagem

2º LISTAS HTML

Não são raros os casos em que queremos exibir uma listagem em nossas páginas. O HTML tem algumas tags definidas para que possamos fazer isso de maneira correta. A lista mais comum é a lista não ordenada definida pela tag <ul> (*unordered list*).

```
<ul>
<li>Primeiro item da lista</li>
<li>
Segundo item da lista:
<ul>
<li>Primeiro item da lista aninhada</li>
<li>Segundo item da lista aninhada</li>
</ul>
</li>
<li>Terceiro item da lista</li>
</ul>
```

Note que, para cada item da lista não-ordenada, utilizamos uma marcação de item de lista <li> (*list item*). No exemplo acima, utilizamos uma estrutura composta na qual o segundo item da lista contém uma nova lista. A mesma tag de item de lista <li> é utilizada quando demarcamos uma lista ordenada.

```
<ol>
<li>Primeiro item da lista</li>
<li>Segundo item da lista</li>
<li>Terceiro item da lista</li>
<li>Quarto item da lista</li>
<li>Quinto item da lista</li>
</ol>
```

As listas ordenadas (<ol> - *ordered list*) também podem ter sua estrutura composta por outras listas ordenadas como no exemplo que temos para as listas não-ordenadas. Também é possível ter listas ordenadas aninhadas em um item de uma lista não-ordenada e vice-versa.

LISTAS DE DEFINIÇÃO

Existe um terceiro tipo de lista que devemos utilizar para demarcar um glossário, quando listamos termos e seus significados. Essa lista é a **lista de definição** (*definition list*).

```
<dl>
<dt>HTML</dt>
<dd>
HTML é a linguagem de marcação de textos utilizada para exibir textos como páginas na Internet.
</dd>
<dt>Navegador</dt>
<dd>
Navegador é o software que requisita um documento HTML
através do protocolo HTTP e exibe seu conteúdo em uma
janela.
</dd>
</dl>
```

Para estilizar o formato padrão das listas ordenadas e não-ordenadas, podemos utilizar a propriedade list-style-type no CSS:

```
ul {
/* alterar para circulo antes de cada <li> da lista não-ordenada */
list-style-type: circle;
}
ol {
/* alterar para uma sequência alfabética antes de cada <li> da lista ordenada */
list-style-type: upper-alpha;
}
```

Também podemos usar a *shorthand property* list-style: circle

LINKS NO HTML

Quando precisamos indicar que um trecho de texto se refere a um outro conteúdo, seja ele no mesmo documento ou em outro endereço (por exemplo uma página na web), utilizamos a tag de âncora <a> . Existem três diferentes usos para as âncoras. Um deles é a definição de links:

```
<p> Visite o site da <a href="https://www.unifran.edu.br">UNIFRAN</a>.
</p>
```

Note que a âncora está demarcando apenas a palavra **UNIFRAN** de todo o conteúdo do parágrafo exemplificado. Isso significa que, ao clicarmos com o cursor do mouse na palavra **UNIFRAN**, o navegador redirecionará o usuário para o site da **UNIFRAN**, indicado no atributo href .

Podemos adicionar o atributo target="" para especificar onde que essa página irá carregar. Por padrão a página irá abrir na mesma aba da página que tem o link, mas se quisermos que a página abra em outra aba podemos colocar o valor _blank dentro desse atributo:

```
<a href="https://www.unifran.edu.br" target="_blank">
```

Outro uso para a tag de âncora é a demarcação de destinos para links dentro do próprio documento, o que chamamos de ***bookmark***.

```
<p>Mais informações <a href="#info">aqui</a>.</p>
<p>Conteúdo da página...</p>
<h2 id="info">Mais informações sobre o assunto:</h2>
<p>Informações...</p>
```

De acordo com o exemplo acima, ao clicarmos sobre a palavra **aqui**, demarcada com um link, o usuário será levado à porção da página onde o *bookmark* **info** é visível. *Bookmark* é o elemento que tem o atributo id.

É possível, com o uso de um link, levar o usuário a um *bookmark* presente em outra página.

```
<a href="https://www.unifran.edu.br/graduacao/ciencia-da-computacao/>
Entre em contato sobre o curso
</a>
```

O exemplo acima fará com que o usuário que clicar no link seja levado à porção da página indicadano endereço, especificamente no ponto onde o *bookmark* **contato** seja visível.

O outro uso para a tag de âncora é a demarcação de destinos para links dentro do próprio site, mas não na mesma página que estamos. Por exemplo, estamos na página **sobre.html** e queremos um link para a página **index.html**.

```
<p>Acesse <a href="index.html">nossa loja</a>.</p>
```

OBJETIVO 2:

Edite seu *.html, insira listas e links direcionando para a sua própria pagina e para links externos.

3º SELETORES MAIS ESPECÍFICOS E HERANÇA

CSS:

```
img {
width: 300px;
}
```

No código acima estamos aplicando uma largura de 300px para todas as tags <img> . Mas e se nós só quisermos aplicar essa largura apenas para as imagens que estão nas figuras? É aí que entra o seletor mais específico:

```
figure img {
width: 300px;
}
```

Agora estamos aplicando a largura de 300px apenas às imagens que são filhas de uma tag <figure> .

Outra forma de selecionar elementos mais específicos é usando o atributo id="" nos elementos que queremos estilizar e depois fazer a chamada de seletor de id :

HTML:

```
<img src="img/xxxxxxxx" atl="xxxxxxxxxx">
<figure>
<img src="img/xxxxxxxx" alt="Foto da xxxxxxxx" id="xxxxxxxxx">
<figcaption>Matriz da MusicDot</figcaption>
</figure>
<figure>
<img src="img/xxxxxxxxx" alt="Foto da xxxxxxxxx" id="xxxxxxxxxx">
<figcaption>Família Tüpfeln</figcaption>
</figure>
```

CSS:

```
#xxxxxxxxxxxx {
width: 300px;
}
#xxxxxxxxxxxx {
width: 300px;
}
```

Só que não é recomendado o uso de id para a estilização de elementos já que a idéia do atributo épara fazer uma referência única na página como fizemos na parte dos links. Quando queremos estilizarelementos específicos é melhor utilizar o atributo class="" . O comportamento no CSS será idênticoao do atributo id="" , mas class foi feito para ser usado no CSS e no JavaScript.

Arrumando o exemplo anterior, usando classes:

HTML:

```
<img src="img/xxxxxxxx" atl="xxxxxxxxxx">
<figure>
<img src="img/xxxxxxxx" alt="Foto da xxxxxxxx" class="xxxxxxxxx">
<figcaption>Matriz da MusicDot</figcaption>
</figure>
<figure>
<img src="img/xxxxxxxxx" alt="Foto da xxxxxxxxx" class="xxxxxxxxxx">
<figcaption>Família Tüpfeln</figcaption>
</figure>
```

CSS:

```
.xxxxxxxxxxxx {
width: 300px;
}
.xxxxxxxxxxxx {
width: 300px;
}
```

Grau de especificidade de um seletor

Existe uma coisa muito importante no CSS que precisamos tomar cuidado é o **grau de especificidade de um seletor**. Isto é, a prioridade de interpretação de um seletor pelo navegador. Para entender estas regras de especificidade de um selector, ao criarmos um seletor de tag a sua pontuação se torna **1**. Quando usamos um seletor de classe sua pontuação se torna **10**. Quando usamos um seletor de id sua pontuação se torna **100**. Ao fim, o navegador soma a pontuação dos seletores aplicados à um elemento, e as propriedades com o seletor de maior pontuação são as que valem.

```
<body>
<p class="paragrafo" id="paragrafo-rosa">Texto</p>
</body>
p { /* Pontuação 1 */
color: blue;
}
.paragrafo { /* Pontuação 10 */
color: red;
}
#paragrafo-rosa { /* Pontuação 100 */
color: pink;
}
```

No exemplo acima o parágrafo vai ficar com a cor rosa porque o seletor que tem a cor rosa é o seletor de maior pontuação.

Quando elementos possuem a mesma pontuação quem prevalece é a propriedade do último seletor:

```
p { /* Pontuação 1 */
color: blue;
}
p { /* Pontuação 1 */
color: red;
}
```

No exemplo acima a cor do parágrafo será vermelha.

Podemos também somar os pontos para deixar nosso seletor mais forte:

```
body p { /* Seletor de tag + outro seletor de tag = 2 pontos */
color: brown;
}
p { /* Pontuação 1 */
color: blue;
}
```

No exemplo acima nós deixamos nosso seletor mais específico para os `<p>` que estão dentro de uma tag `<body>`, portanto a cor do parágrafo será marrom.

Em resumo:
Quanto mais específico é o nosso seletor, maior sua pontuação no nível de especificidade do CSS.
Portanto devemos sempre trabalhar com uma baixa especificidade, para que não seja impossível sobrescrever valores quando necessário em uma situação específica.

Herança

A cascata do CSS, significa justamente a possibilidade de elementos filhos herdarem características de estilização de elementos superiores, estas definidas por suas propriedades, que podem ou não passar aos seus descendentes seus valores.
Vamos ver o exemplo de código a seguir:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Um exemplo</title>
</head>
<body>
<p>Uma breve explicação de algo com um <a href="https://google.com.br">link</a> para uma referência de outra página</p>
<figure>
<img src="img/foto" alt="Uma foto">
<figcaption>Uma foto</figcaption>
</figure>
</body>
</html>
```

Vamos mudar a família da fonte de toda a página. Uma maneira que podemos fazer é selecionar todas as tags que contém text (<p> , <a> e <figcaption>) e colocar a família de fonte que queremos:

```
p {
font-family: 'Helvetica', sans-serif;
}
a {
font-family: 'Helvetica', sans-serif;
}
figcaption {
font-family: 'Helvetica', sans-serif;
}
```

Mas isso dá muito trabalho e estamos repetindo código. Ao invés de colocar essa propriedade em cada um dos elementos textuais da nossa página, podemos colocar no elemento superior a estas tags, neste caso é o elemento <body> .

```
body {
font-family: 'Helvetica', sans-serif;
}
```

No exemplo acima todos os elementos filhos da tag <body> vão receber a propriedade fontfamily:

e isso é o que nós chamamos de **herança**. Herança acontece quando elementos herdam propriedades dos elementos acima deles (elementos pai).

Obs: Para saber se uma propriedade deixa herança ou não, é possível consultar na sua especificação ou no site MDN https://developer.mozilla.org/.

PARA SABER MAIS: O VALOR INHERIT

Imagine que temos a seguinte divisão com uma imagem:

```
<div>
<img src="box-model.png" alt="box model">
</div>
div {
border: 2px solid;
border-color: red;
width: 30px;
height: 30px;
}
```

Queremos que a imagem preencha todo o espaço da , mas as propriedades width e height não são aplicadas em cascata, sendo assim, somos obrigados a definir o tamanho da imagem manualmente:

```
img {
width: 30px;
height: 30px;
}
```

Esta não é uma solução sustentável, porque, caso alterarmos o tamanho da , teremos que lembrar de alterar também o tamanho da imagem. Uma forma de resolver este problema é utilizando o valor **inherit** para as propriedades width e height da imagem:

```
img {
width: inherit;
height: inherit;
}
```

O valor inherit indica para o elemento filho que ele deve utilizar o mesmo valor presente no elemento pai, sendo assim, toda vez que o tamanho do elemento pai for alterado, automaticamente o elemento filho herdará o novo valor, facilitando assim, a manutenção do código.
Lembre-se de que o inherit também afeta propriedades que não são aplicadas em cascata.

OBJETIVO 3:

Nesse exercício, diminuiremos a largura das duas tags <figure> da página. Além das larguras, também alteraremos a disposição de cada figura na página. Cada uma terá uma largura e disposição diferentes.
A <figure> da matriz deverá ter **550px** de largura e deve estar centralizada em relação à página. A <figure> da família devera ter **275px** de largura, deve ficar alinhada à direita e envolta pelos textos que no HTML vêm abaixo dela.
Ao alterar a largura das tags <figure> você notará que as tags <img> dentro delas são maiores e ultrapassam a caixa da figura Para resolver isso podemos dizer que toda tag <img> dentro de uma tag <figure> deve ocupar todo o espaço que definimos, mas não mais que isso.