**WORLD WIDE WEB**

**World Wide Web**

A *World Wide Web* (Rede de Alcance Mundial) é uma rede de conteúdo online formatado em HTML e acessado via HTTP através de um *browser*;

De forma mais simples, é um sistema de documentos em *hypermedia* que são interligados e executados na internet, que acessamos com os navegadores.

**Hypermedia e Hyperlink**

***Hypermedia*:** são múltiplas formas de mídia (imagens, sons, vídeos, textos etc) contidas em *webpages* e que carregam consigo *hyperlinks*;

***Hyperlink* (ou link):** é a conexão que permite o usuário se mover entre *webpages*, documentos, mídia digital etc.

De forma mais básica, é a parte “invisível” que ocorre com a *hypermedia*, isto é, quando clicamos é uma imagem com um *link* (*hyperimage*) que nos leva a um vídeo, o que faz essa conexão da imagem com o vídeo é chamado de *hyperlink*. O mesmo se aplica a todos os outros casos (texto para texto, texto para imagem, imagem para imagem, imagem para som etc).

**OBS:** “*hypermedia*” não se confunde com “*multimedia*”, o qual se define como uma forma de comunicação visual que utiliza uma combinação de imagens, sons e palavras, sejam elas estáticas ou não.

Ou seja, um smartphone é um “aparelho *multimedia*”, já que possui imagens estáticas, imagens se movendo (vídeos), sons, combinações de sons (música) e palavras e texto no geral.

Apenas um destes elementos sozinhos se caracterizaria como simplesmente “*media*”, e assim, quando você escuta uma música no seu celular, você está escutando uma “*media*” no seu aparelho *multimedia*.

**HTTP**

HTTP significa *HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto);

É um protocolo de comunicação de *request-response* (requisição e resposta) utilizado pela WWW para definir como as mensagens são formatadas e transmitidas e quais ações os servidores e navegadores devem realizar em resposta a vários comandos;

De forma simplificada, é a conexão para que o *client-server* (usuário e servidor) se comuniquem;

Quando você clica em um vídeo na página inicial do Youtube, você acessa este vídeo por meio do protocolo HTTP (e também graças à sua internet, obviamente).

**OBS:** HTTPS (*HyperText Transfer Protocol Secure*) é a mesma coisa que HTTP, porém a conexão é encriptada por TLS (*Transport Layer Security*), que é um protocolo de segurança cuja finalidade é facilitar a segurança e privacidade de dados de dados na internet.

**INTRODUÇÃO AO HTML**

**HTML**

HTML significa *Hyper Text Markup Language* (Linguagem de Marcação de Hipertexto);

É uma linguagem de marcação (não é uma linguagem de programação);

É utilizada para definir a estrutura de páginas da web;

Consiste em uma série de *elements* que dizem para o navegador como mostrar o conteúdo (título, corpo do texto, parágrafo, linha sem quebra de texto, a fonte da letra, as cores etc).

***Element* de HTML**

Um *element* de HTML é definido por uma tag inicial, seu conteúdo interno e uma tag final;

Um *element* HTML pode ser *nested* (aninhado), isto é, pode haver um elemento dentro de outro.

|  |  |  |
| --- | --- | --- |
| ***Element* de HTML** | | |
| **Tag inicial** | **Conteúdo** | **Tag final** |
| <tagname> | Tag content | </tagname> |

***Attribute* de HTML**

Um *attribute* de HTML é um complemento para o elemento, provendo informações adicionais para tal;

Um *atribute de* HTML sempre deve ser declarado no início da tag, e funcionam comumente como nomedoatributo=”valor”

|  |  |  |
| --- | --- | --- |
| ***Attribute* de HTML** | | |
| **Abrindo a tag** | **Definindo um *attribute*** | **Fechando a tag** |
| <tagname | atribute=”value”  (pode conter mais de um atributo) | >  ou  Tag content</tagname> |

**OBS¹:** os valores escritos dentro dos atributos não precisam ser abertos e fechados com aspas, porém, por boa prática, é recomendável coloca-las para funcionar em documentos mais restritos como os de XHTML.

**OBS²:** os valores escritos dentro dos atributos podem ser escritos com aspas duplas ou aspas simples, contudo, caso um valor de um atributo precise ter aspas duplas ou simples dentro, logo é necessário o uso de ambas, não importando sua ordem.

Exemplo:

title=”Meu ‘novo’ título”

title=’Meu “novo” título’

**Estrutura básica de um documento em HTML**

**<!DOCTYPE html>** é a declaração que define que o documento é um documento em HTML5. Todos os documentos em HTML devem ser declarados com esta tag antes de qualquer tag. Ainda, todo documento de HTML não é *case-sensitive*

**<html>** é o elemento raiz de uma página em HTML

**<head>** é o elemento que contém a metainformação da página em HTML

**<title>** é o elemento que define o título da página em HTML

**<body>** é o elemento que define todo o corpo do documento, é o container que abrigará todo o conteúdo visual da página

**lang** é um atributo para definir qual o idioma da webpage, sua utilidade se dá principalmente a respeito de *search* *engines* (mecanismos de pesquisa)

Código:

<!DOCTYPE html>

<html lang=”pt-BR”>

<head>

<meta charset="utf-8">

<title>Título</title>

</head>

<body>

TODO A ESTRUTURA DA PAGE FICA AQUI

</body>

</html>

**OBS¹:** *case-sensitive* significa “sensível a maiúsculas e minúsculas”. Ou seja, aquilo que não é *case-sensitive* não precisa diferenciar maiúsculas ou minúsculas para funcionar corretamente.

**OBS²:** mesmo que o HTML não seja *case-sensitive*, é recomendável pela própria W3C que as tags e atributos sejam escritas em lowercase (caixa baixa), para assim funcionar em documentos mais restritos como os de XHTML.

**HTML: TAGS BÁSICAS**

**Tag de título**

Tags de título são chamadas de tags de *Heading*. As tags de título já vêm por padrão em *bold* (negrito) e com uma *line-break* (quebra de linha).

A diferença visual entre cada uma delas é simplesmente o tamanho, porém, se tratando de *search engines* (mecanismos de pesquisa) as headings são fundamentais para indexar a estrutura e o conteúdo das páginas da web. Por isso, apenas use headings para criar títulos, e não para deixar palavras/frases em negrito ou maiores/menores.

Código:

<h1>This is heading 1</h1>

<h2>This is heading 2</h2>

<h3>This is heading 3</h3>

<h4>This is heading 4</h4>

<h5>This is heading 5</h5>

<h6>This is heading 6</h6>

Como fica visualmente:

**This is heading 1 (tamanho de 32px)**

**This is heading 2 (tamanho de 24px)**

**This is heading 3 (tamanho de 20px)**

**This is heading 4 (tamanho de 16px)**

**This is heading 5 (tamanho de 12px)**

**This is heading 6 (tamanho de 10px)**

**Tag de parágrafo e de texto pré-formatado**

**<p>** mostra o texto na tela e coloca uma quebra de linha automática no fim, seu texto não é pré-formatado, portanto, espaços adicionais entre letras ou linhas não são considerados

**<pre>** mostra o texto na tela e coloca uma quebra de linha automática no fim, seu texto é pré-formatado, portanto, espaços adicionais entre letras ou linhas são considerados, e ainda, sua fonte padrão é a Courier

Código:

<p>This is paragraph. I keep writing on the paragraph.</p>

<p>This is paragraph. I

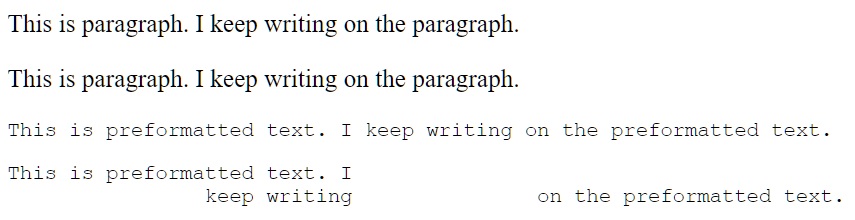
keep writing on the paragraph.</p>

<pre>This is preformatted text. I keep writing on the preformatted text.</pre>

<pre>This is preformatted text. I

keep writing on the preformatted text.</pre>

Como fica visualmente:



**OBS**: parágrafo é diferente de quebra de linha, uma vez que uma quebra de linha não inicia um novo parágrafo.

**Tags de texto padrão e link**

**<span>** mostra o texto na tela, mas não coloca uma quebra de linha no fim

**<a>** criar um hyperlink, direcionando o usuário para o endereço definido dentro do atributo “href”. Assim como a tag <span>, o <a> não coloca uma quebra de linha automática no fim, e também, por padrão, o texto ficará com um *underline* (sublinhado) e ficará na cor azul

Código:

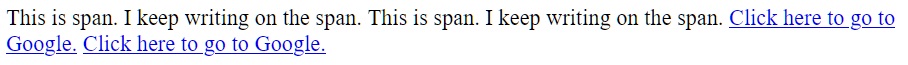
<span>This is span. I keep writing on the span.</span>

<span>This is span. I keep writing on the span.</span>

<a href=”https://www.google.com”>Click here to go to Google.</a>

<a href=”https://www.google.com”>Click here to go to Google.</a>

Como fica visualmente:



**OBS**: tanto o <span> quanto o <a> não adicionam quebras de linha automáticas, por isso tudo acabou ficando colado. Para resolver isso, podemos usar a solução do próximo tema.

**Tags de quebra de linha e divisões**

**<br>** adiciona uma quebra de linha na *page*

**<hr>** adiciona uma quebra de linha na *page,* porém com uma linha separando o conteúdo de cima com o de baixo

**<div>** é uma das formas mais profissionais de se dividir corretamente o documento. O <div> pode ser usado como quebra de linha, porém, não é seu objetivo principal, ele é muito melhor aproveitado e explorado quando se utiliza id e classes em conjunto com o CSS

Código:

<span>This is span. I keep writing on the span.</span>

<br>

<span>This is span. I keep writing on the span.</span>

<hr>

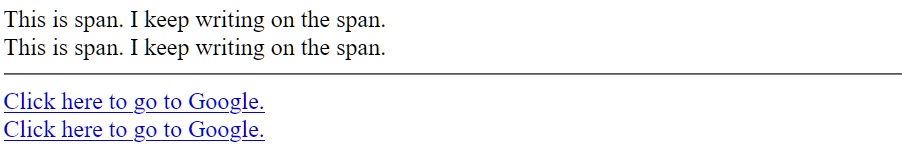
<a href=”https://www.google.com”>Click here to go to Google.</a>

<div>

<a href=”https://www.google.com”>Click here to go to Google.</a>

</div>

Como fica visualmente:



**Tags de imagem**

As imagens são definidas pela tag <img>, junto de seus atributos internos, como “src”, “width” e “height”. A imagem pode ser colocada com uma URL relativa ou absoluta, isto é, pode-se utilizar uma imagem que está na pasta do documento em HTML ou na web.

**<img>** é a tag base para colocar uma imagem

**src** é o *source* (fonte) da imagem, isto é, o nome da sua imagem com sua extensão ou então o link da web onde a foto está hospedada

**alt** caso a imagem não possa ser mostrada por qualquer motivo, o valor (texto) definido neste atributo será mostrado

**title** quando o mouse ficar parado em cima do elemento (neste caso, uma imagem) será mostrada uma caixa de texto com o valor (texto) definido neste atributo

**width** é a largura da imagem

**height** é a altura da imagem

Código:

<img src=”foto.jpg” alt=”foto” title=”imagem” width=”500”>

<img src=”foto.jpg” alt=”foto” title=”imagem” width=”500” height=”500”>

<img src=”https://www.imgutr.com/W5afT” alt=”foto” title=”imagem” width=”100” height=”500”>

Como fica visualmente:



 Imagem com URL relativa (o texto não é visível) – *width* = 500 / *height* = auto

Imagem com URL relativa (o texto não é visível) – *width* = 500 / *height* = 500



Imagem com URL absoluta (o texto não é visível) – *width* = 100 / *height* = 500

**OBS¹:** se você definir apenas um atributo de tamanho (largura ou altura) a imagem se ajustará automaticamente para não causar distorções na imagem (olhar imagem sem requisição 1).

**OBS²:** a tag de <img> não precisa de uma tag fechando-a, como </img>.

**Tags de tabela**

Para as tabelas são usadas diversas tags, algumas são apenas para organização, enquanto outras de fato possuem suas particularidades visuais e organizacionais. A tag “table” compreende atributos presentes na <img>, como “width” e “height”, além também do atributo “border”.

**<table**> é a tag padrão para criação de tabelas

**border** atributo que adiciona uma borda na tabela, sendo este valor definido numericamente. O número neste atributo definirá a espessura da borda

**<thead> e <tbody>** são utilizadas apenas para a organização de tabelas, sendo uma para organizar as tags do head (título) da tabela, e a outra para organizar as tags do body (corpo) da tabela

**<tr>** é utilizada para definir cada linha da tabela

**<th>** é utilizada para definir cada titulo de cada coluna. O que estiver escrito nesta tag virá *centered* (centralizado) e em bold (negrito) por padrão

**<td>** é utlizada para definir os valores/células na tabela. O que estiver escrito nesta tag virá *left-aligned* (alinhado à esquerda)

Código:

<table width=”1280” border=”1”>

<thead>

<tr>

<th>NOME</th>

<th>IDADE</th>

<th>CARGO</th>

</tr>

</thead>

<tbody>

<tr>

<td>Marcos</td>

<td>40</td>

<td>Advogado</td>

</tr>

<tr>

<td>Maria</td>

<td>28</td>

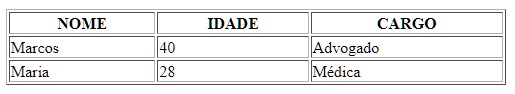
<td>Médica</td>

</tr>

</tbody>

</table>

Como fica visualmente:



**Tags de formulário**

Aqui são compreendidas tags utilizadas para a criação e envio de formulários na web. Foram colocadas tags <br> no código para adicionar quebras de linha, uma vez que essas tags não fazem isso automaticamente.

**<form>** é a tag base de formulário

**<input>** é utilizada para adicionar caixas de texto onde é possível escrever

**type** é utilizado para definir o tipo do input, podendo ser “text”, “password“, “button” ou “submit”

**type=”text”** o texto dentro da caixa ficará normal

**type=”password”** o texto dentro da caixa ficará todo em asterisco (\*)

**type=”button”** transforma a caixa de texto em um botão que não envia um formulário para o site quando clicado. Esse tipo de input não faz nada por padrão, e são comumente usados para atuar com o JavaScript ou AJAX

**type=”submit”** transforma a caixa de texto em um botão que envia um formulário para o site, é a função padrão dele (é o botão usado para salvar o login de um site, por exemplo). Isso é melhor compreendido quando se estuda Back-End

**name** é usado apenas para envio de valores no Back-End, nada interessa ao Front-End

**placeholder** é o texto que ficará escrito quando não houver nada escrito na caixa de texto

**value** é o texto que ficará escrito no botão, seja ele de tipo “button” ou “submit”

**textarea** é utilizado para criar uma caixa de texto da qual é possível redimensiona-la em tempo real

Código:

<form>

<input type="text" name="" placeholder="Digite seu texto">

<br>

<input type="password" name="" placeholder="Digite sua senha">

<br>

<input type="button" name="" value="Enviar">

<br>

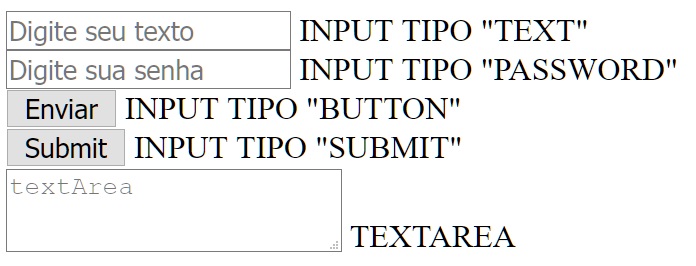
<input type="submit" name="" value="Submit">

<br>

<textarea placeholder="textArea"></textarea>

</form>

Como fica visualmente:



O texto ao lado de cada tag não é visível na page

**Tags de lista**

**<ul>** é a tag base para criar uma lista *unordered* (não ordenada)

**<ol>** é a tag base para criar uma lista *ordered* (ordenada)

**<li>** é a tag utilizada para criar uma *line* (linha) dentro da lista, isto é, serve para criar um novo item

**<dl>** é a tag base para criar uma lista de *description* (lista de descrição)

**<dt>** é a tag utilizada para adicionar uma linha que conterá algum termo para ser descrito

**<dd>** é a tag utilizada para descrever o termo dentro da tag <dt>

Código:

<ul>

<li>Café</li>

<li>Chá</li>

<li>Leite</li>

</ul>

<ol>

<li>Café</li>

<li>Chá</li>

<li>Leite</li>

</ol>

<dl>

<dt>Café</dt>

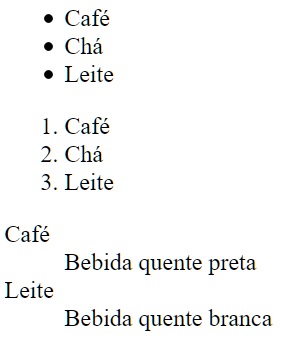
<dd>Bebida quente preta</dt>

<dt>Leite</dt>

<dd>Bebida quente branca</dt>

</dl>

Como fica visualmente:



**OBS:** basicamente a diferença entre os três tipos de lista é: a *unordered list* não ordena os itens por números (apenas coloca uma bola na frente de cada), a *ordered list* ordena os itens por números e a *description list* não coloca uma bola ou número na frente do item, mas a descrição de cada item é automaticamente colocado um tabulamento para a direita.

**Tags de formatação de texto**

**<b>** é a tag para apenas deixar o texto em negrito

**<strong>** é a tag para deixar o texto em negrito e também o marcar como importante

**<i>** é a tag para apenas deixar o texto em itálico

**<em>** é a tag para deixar o texto em itálico e também indicar que é um termo técnico, uma palavra/frase de outra língua, um pensamento, isto é, para enfatizar a palavra

**<mark>** é a tag para deixar o texto marcado com um fundo amarelo

**<small>** é a tag para deixar o texto menor do que ele é por padrão

**<del>** é a tag para deixar o texto com uma linha o cortando pela metade. É usado para colocar uma palavra como “removida” do texto

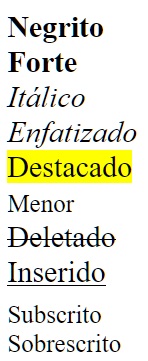
**<ins>** é a tag para deixar o texto em sublinhado. É usado para colocar uma palavra como “inserida” no texto

**<sub>** é a tag para deixar o texto subscrito, isto é, ele fica com a metade do tamanho que estava anteriormente para baixo da metade. É usado para escrever, por exemplo, fórmulas químicas

**<sup>** é a tag para deixar o texto sobrescrito, isto é, ele fica com a metade do tamanho que estava anteriormente para cima da metade. É usado para escrever, por exemplo, potenciação

Código:

Como fica visualmente:



**OBS¹:** nenhuma tag de formatação adiciona uma quebra de linha automática, isso foi colocado no visual apenas para melhor visualização

**OBS²:** é possível colocar todas essas tags dentro de outra tag de texto, como por exemplo, colocar a tag de subscrito dentro de apenas uma letra de um span.

Exemplo: <span>A fórmula química da água é H<sub>2</sub>O.</span>

**INTRODUÇÃO AO CSS**

**CSS**

CSS significa Cascading Style Sheets (Folha de Estilos em Cascata);

É uma linguagem de estilização (não é uma linguagem de programação);

É usada para estilizar páginas que foram estruturadas com o HTML;

Com o CSS é possível estilizar o mesmo código em HTML de diversas formas diferentes.

**OBS¹:** o HTML foi criado apenas com o intuito de mostrar o conteúdo das páginas da web, e não para formata-la de formas diferentes. Quando o HTML 3.2 introduziu tags de formatação de texto foi um grande pesadelo para os desenvolvedores, pois para grandes sites a formatação tinha de ser feita em cada uma das páginas, o que era um processo muito longo e repetitivo. E assim, para resolver este problema, foi criado o CSS pela W3C. Dessa forma, toda a formatação da página agora ficava encarregada do CSS, deixando o HTML apenas com a função de construir a estrutura básica do site.

**Regras, seletores, declarações, propriedades e valores**

***1) Rule* (regra) de CSS:** consiste em um *selector* (seletor) e um *declaration block* (bloco de declaração), sendo que dentro deste ficarão as *properties* (propriedades) e os *values* (valores). Ela possui uma sintaxe própria que define como será aplicado estilo a um ou mais elementos da marcação HTML de uma página. Um conjunto de regras CSS formam uma folha de estilos.

**1.1) Selector (seletor):** é um elemento da marcação HTML identificado pelo nome da tag (<p>, <h1>, <form>, etc), classe (topo, .principal, .menu, etc), identificador (#tudo, #auxiliar, #rodape, etc), conjunto de tags, classes e identificadores, ou asterisco (\* seleciona todo o documento HTML). O *selector* é dividido em cinco categorias:

**1.1.1)** **Simple Selector (seletor simples):** seleciona elementos com base no nome da tag, classe ou id;

**1.1.2)** **Combinated Selector (seletor combinado):** seleciona os elementos com base em uma relação específica entre eles;

**1.1.3) Pseudo-class Selector (seletor de pseudo-classe**): seleciona os elementos com base em uma determinada situação;

**1.1.4) Pseudo-elements Selector (seletor de pseudo-elementos):** seleciona apenas parte de um elemento;

**1.1.5) Attribute Selector (seletor de atributo):** seleciona elementos com base em um atributo ou valor de um atributo.

**1.2) Declaration Block (blodo de declaração):** é o conjunto de uma ou mais propriedades e um valores, aqui é onde ocorre a estilização das marcações de HTML.

**1.2.1) Property (propriedade):** é a propriedade do elemento HTML ao qual será aplicada a estilização definida no valor;

**1.2.2) Value (valor):** é a característica específica a ser assumida pela propriedade.

|  |  |  |
| --- | --- | --- |
| ***Rule* de CSS** | | |
| ***Selector*** | ***Declaration*** | |
| h1 | {font-size: | 12;} |
|  | *Property* | *Value* |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Rule* de CSS com mais de um *selector* e *declaration*** | | | | |
| ***Selector*** | ***Declaration*** | | ***Declaration*** | |
| h1, .classe1 p | {font-size: | 12; | color: | red;} |
|  | *Property* | *Value* | *Property* | *Value* |

**OBS¹:** é possível utilizar mais de uma *declaration* dentro de uma *rule*, assim possibilitando estilizar mais de uma propriedade de um *selector*. Para isso, basta colocar um ; após cada *declaration*, e então prosseguir escrevendo a próxima *declaration*. Exemplo: h1 {font-size:12; color:blue;}

**OBS²:** caso o valor da propriedade seja uma palavra composta, esta deverá estar entre aspas simples ou duplas, como no caso de definir o tipo de fonte Exemplo: h1 {font-family:”Comic Sans MS”;)

**OBS³:** é recomendável sempre colocar um ; após a última *declaration*, para no caso de futuramente ter que se acrescentar mais declarações na regra de CSS (evitando correr o risco de esquecer eventual ponto-e-vírgula inicialmente omitido).

**Formas de utilizar o CSS em um documento HTML**

Há três formas diferentes para se usar o CSS dentro de um documento em HTML, são elas:

**1) CSS Inline:** a estilização será feita na mesma linha da tag que se pretende estilizar. Para fazer isso, basta abrir uma tag qualquer e dentro dela digitar:

<tagname “style=propriedade1:valor1;propriedade2:valor2;”>Tag contente</tagname>

Código do .HTML:

<!DOCTYPE html>

<html lang=”pt-BR”>

<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>

<p style="color:red;">This is a paragraph.</p>

</body>

</html>

**2) CSS internal:** a estilização será feita no documento HTML sendo estilizado, sendo que toda a configuração do CSS ficará no início do documento. Para fazer isso, basta abrir uma tag <style> logo após a tag <head> e começar a estilizar o documento o CSS:

Código do .HTML:

<!DOCTYPE html>

<html lang=”pt-BR”>

<head>

<style>

body {

background-color: linen;

}

h1 {

color: blue;

margin-left: 40px;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

**3) CSS external:** a estilização será feita em um documento próprio com a extensão “.css”, e toda a configutração do CSS ficará dentro deste documento, sendo que no documento HTML apenas será feita a linkagem com o documento “.css”. Para fazer isso, é necessário criar uma tag <link> no documento de HTML e então colocar os atributos “rel”, “type” e "href”:

Código do .HTML:

<!DOCTYPE html>

<html lang=”pt-BR”>

<head>

<link rel="stylesheet" type="text/css" href="cstyle.css">

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>

Código do .CSS:

body {

background-color: lightblue;

}

h1 {

color: navy;

margin-left: 20px;

}

**OBS¹:** o atributo “rel” é para identificar que o documento sendo linkado é um documento .css; o atributo type é opcional, apenas é colocado por boa prática; o atributo href é o caminho onde está localizado o documento .css, sendo que o caminho de origem será onde o documento .html se encontra.

**OBS²:** é possível utilizar as 3 maneiras distintas de se chamar o CSS em um mesmo documento .html.

**OBS³:** quando há mais de um estilo igual sendo especificado dentro de um mesmo documento .html, haverá maior prioridade para a forma “inline”.

**Cores**

Há várias formas para definir uma cor no CSS. Diversos elementos podem ter suas cores editadas, como texto, background, bordas etc. Para modificar as cores podemos utilizar valores em:

**1) RGB:** rgb(255,0,0)

**2) RGBA:** rgba(0,0,255,0.5)

**3) HEX:** #d261ff

**4) HSL:** hsl(200, 100%, 50%)

**5) HSLA:** hsla(66, 100%, 69%, 0.55)

**6) Name:** green

Código do .HTML:

<!DOCTYPE html>

<html lang=”pt-BR”>

<head>

<link rel="stylesheet" type="text/css" href="cstyle.css">

<body>

<h1>Heading 1</h1>

<h2>Heading 2</h2>

<h2>Heading 3</h2>

<h2>Heading 4</h2>

<h2>Heading 5</h2>

<h2>Heading 6</h2>

</body>

</html>

Código do .CSS:

h1 {

color: rgb(255,0,0);

}

h2 {

color: rgba(0,0,255,0.5);

}

h3 {

color: #d261ff;

}

h4 {

color: hsl(200,100%,50%);

}

h5 {

color: hsla(66,100%,69%,0.55);

}

H6 {

color: lightblue;

}

**OBS¹:** não há uma *declaration* específica para o texto, como por exemplo “font-color”. O correto, para modificar a cor do texto, é apenas escrever “color”. Porém, outros elementos possuem certas *declarations* específicas, como “background-color” e “border-color”.

**Backgrounds**

**1) background-color:** é utilizada para definir a cor de fundo de um elemento.

body {

background-color:green;

}

**2) background-image:** é utilizada para definir uma imagem como fundo de algum elemento. É necessário um atributo “url” para especificar qual será a imagem que ficará de fundo. Ainda, por padrão, a imagem é repetida no eixo horizontal e vertical, isto é, quando a imagem acaba, ela começa de novo no fundo.

body {

background-image:url(photo.jpg);

}

**3) background-repeat:** é utilizada para definir se a imagem irá se repetir ou não. É possível utilizar os valores:

**a) “repeat-x”:** a imagem será repetida apenas na horizontal;

**b) “repeat-y”:** a imagem será repetida apenas na vertical;

**c) no-repeat:** a imagem não será repetida;

body {

background-image:url(photo.jpg);

background-repeat:repeat-x;

/\*background-repeat:repeat-y;

background-repeat:no-repeat;\*/

}

**4) background-position:** altera a posição padrão do background. É possível utilizar diversos valores de medidas aqui, desde os padrões (top, bottom, top left, bottom right, center etc) até os mais complexos (cm, em, ch, %, etc)

body {

background-image:url(photo.jpg);

background-position:top right;

}

**5) background-attachment:** é utilizada para definir se a imagem ficará fixa ou não quando se rola o scroll do mouse.

body {

background-image:url(photo.jpg);

background-attachment:fixed;

/\*background-attachment:scroll;\*/

}

**6) background:** quando apenas é escrito background, sem maiores detalhes, é possível o detalhar com vários valores em apenas uma linha, sem necessidade de escrever diversas declararações com seus respectivos valores. Este tipo de declaração é chamada de *“shorthand property”.*

body {

background: #ffffff url("img\_tree.png") no-repeat right top;

}

**OBS¹:** é possível alterar a cor de fundo de qualquer elemento, seja um parágrafo (<p>), um título (<h1>) ou toda a página (<body>) etc.

**OBS²:** é possível alterar a opacidade de elementos com a declaração “opacity”.

**Bordas**

É possível modificar diversos aspectos da propriedade “border”, como estilo, tamanho e cor.

**1) border-style:** é utilizada para definir o estilo da borda. Há diversos estilos de borda, sendo eles: dotted, dashed, solid, double, groove, ridge, inset, outset, none e hidden. O estilo padrão é o solid.

h1 {

border-style: dotted;

/\* border-style: dotted solid groove double; É POSSÍVEL DEFINIR OS 4 LADOS DA BORDA, EM

SENTIDO HORÁRIO

border-top-style: dotted;

border-right-style: double;

border-bottom-style: ridge;

border-right-style: dashed; É POSSÍVEL FAZ A MESMA COISA SEPARADAMENTE TAMBÉM \*/

}

**2) border-width:** é utilizada para definir a grossura da borda. Esta propriedade aceita diversas medidas diferentes (px, pt, cm, em, etc), e também possui algumas padrões como *thin* (fina), *medium* (média) *e thick* (grossa). A grossura padrão é de 1px.

h1 {

border-style: solid;

border-width: 5px; É POSSÍVEL FAZ A MESMA COISA PARA CADO LADO TAMBÉM, EM LINHAS DIFERENTES.

ISTO É, DEFINIR UMA LINHA COM 4 VALORES DIFERENTES COMO NO BORDER-STYLE NÃO FUNCIONA \*/

}

**3) border-color:** é utilizada para definir a cor da borda. Todos os valores de cor são aceitos aqui (rgb, rgba, hex, hsl, hsla e name). A cor padrão da borda é preto.

h1 {

border-style: solid;

border-color: rgb(35,87,188);

/\*border-color: red green blue yellow; É POSSÍVEL DEFINIR A COR DOS 4 LADOS DA BORDA, EM

SENTIDO HORÁRIO \*/

}

**4) border:** quando apenas é escrito border, sem maiores detalhes, é possível o detalhar com vários valores em apenas uma linha, sem necessidade de escrever diversas declararações com seus respectivos valores. Esta é a *“shorthand property”* para as bordas*.*

h1 {

border: 3px dotted red;

}

**5) border-radius:**

h1 {

border: 10px solid green;

border-radius: 5px;

}

**Margens**

As margens no CSS são utilizadas para criar espaço em volta dos elementos, fora de qualquer borda.

**margin:** define a margem para o elemento.

h1 {

margin: 10px;

margin: 10px 50px; É POSSÍVEL DEFINIR AS MARGENS COM 2 VALORES, FAZENDO QUE SE DEFINA UM

VALOR PARA AS MARGENS DE CIMA E BAIXO NO PRIMEIRO VALOR E ESQUERDA E DIREITA NO SEGUNDO.

margin: 10px 20px 40px 70px; É POSSÍVEL TAMBÉM DEFINIR OS 4 VALORES DISTINTAMENTE NA MESMA

LINHA OU EM LINHAS SEPARADAS

margin-top: 10px;

margin-right: 20px;

margin-bottom: 40px;

margin-left: 70px;

}

**OBS¹:** um valor muito usado para margens é o valor “auto”, que basicamente significa “automático”, e sua função em margens costuma ser para centralizar algum elemento. Isto é, colocado um valor apenas para as margens superior e inferior, e então o valor dos lados é deixado como “auto”, para assim o valor da margem da esquerda e direita serem idênticos, fazendo com que o elemento fique horizontalmente centralizado, sem precisar quebrar a cabeça para saber onde é o meio do documento html.

**OBS²:** é possível ocorrer um *margin* *collapse* (colapsso de margens) com as margens superior e inferior, apenas (não ocorre com as margens laterais). Quando um elemento possui uma margem inferior de 70px e outro elemento logo abaixo uma margem superior de 20px, tecnicamente o segundo elemento deveria ir mais para baixo, dando 20px de espaço da margem de 70px do primeiro elemento, resultando assim em uma distância de 90px do elemento 1 ao elemento 2. Contudo, não é isso que ocorre, pois o elemento 2 irá criar uma margem de 20px para cima, literalmente entrando dentro da margem do elemento 1, e portando, o elemento 2 não irá descer 20px, mesmo que tenha especificado uma margem de 20px superior.

**Espaçamento interno**

O espaçamento interno no CSS é chamado de *padding*, e com ele criamos espaço entre o conteúdo do elemento, dentro de quaisquer bordas definidas.

**padding:** define o espaçamento interno de um elemento.

h1 {

padding: 10px;

padding: 10px 50px; É POSSÍVEL DEFINIR O PADDING COM 2 VALORES, FAZENDO QUE SE DEFINA UM

VALOR PARA O PADDING DE CIMA E BAIXO NO PRIMEIRO VALOR E ESQUERDA E DIREITA NO SEGUNDO.

padding: 10px 20px 40px 70px; É POSSÍVEL TAMBÉM DEFINIR OS 4 VALORES DISTINTAMENTE NA MESMA

LINHA OU EM LINHAS SEPARADAS

padding-top: 10px;

padding-right: 20px;

padding-bottom: 40px;

padding-left: 70px;

}

**OBS:** o padding adiciona mais pixels para os elementos. Isto é. Se eu defini que meu elemento tem 300px de largura, quando eu aplicar um padding de 50px em cada lado, logo ele passará a ter 400px no total, uma vez que o padding não colapse com o tamanho já pré-definido do elemento. Caso não seja a ideia aumentar o tamanho do elemento, podemos usar a declaração “box-sizing: border-box;”. Está propriedade será estudada na sua respectiva seção.

**Contorno**

Um *outline* (contorno) é uma linha que é desenhada fora da *border* do elemento, ou seja, é possível utilizar a *outline* junto de uma *border*.

A grande diferença de uma *outline* para uma *border* é que a *outline é desenhada fora do elemento e pode sobrepor* outros conteúdos, e ainda, a *outline* não faz parte das dimensões do elemento, isto é, a altura e largura do elemento não são afetadas por ela.

As *outlines* possuem diversas propriedades, sendo elas:

**outline-style:** define o estilo do contorno (possui os mesmos estilos da *border*);

**outline-color:** define a cor do contorno;

**outline-width:** define a grossura do contorno;

**outline-offset:** define uma distância entre o contorno e a borda do elemento;

**outline:** é a “*shorthand* *property*” para os contornos, sendo que dentro dela é colocado seu tamanho, seu estilo e sua cor, especificamente nesta ordem (o *outline-offset* não é declarado dentro da *shorthand property*).

h1 {

outline: 10px solid black;

outline-offset: 6px;

outline-style: dashed;

outline-color: powerblue;

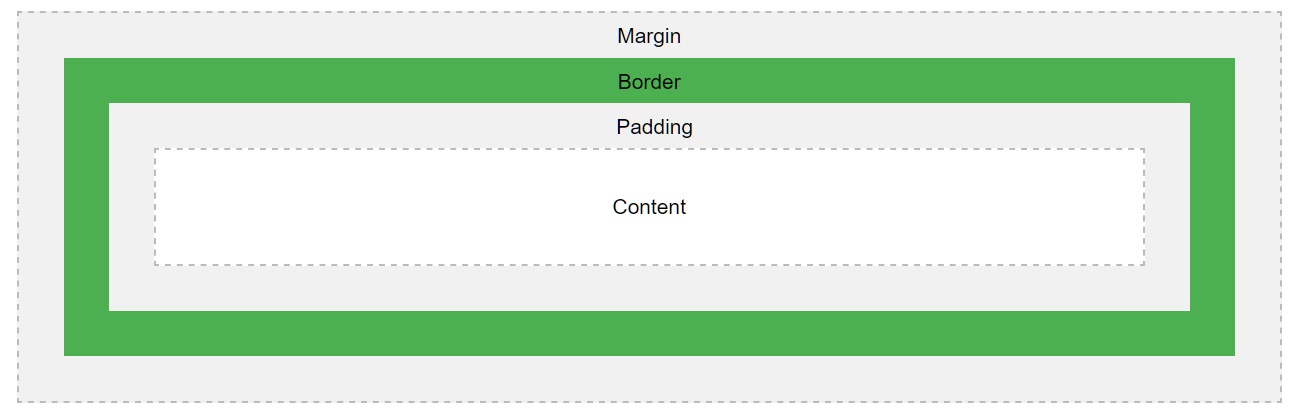
outline-width: 5px;

}

**Modelo de caixa**

A *box* *model* (modelo de caixa) no CSS é utilizada quando estamos falando de design e layout de uma page.

A *box model* é uma caixa que se envolve em todos os elementos de HTML, sendo constituída de: margens, bordas, espaçamentos internos e o conteúdo do elemento. A imagem abaixo ilustra o citado:



**1) Content:** é o conteúdo em si do elemento HTML;

**2) Padding:** é o espaçamento interno do elemento, a área em volta do conteúdo, sendo ele transparente;

**3) Border:** é a borda que fica entre o elemento HTML e seu espaçamento interno (caso exista alguma *padding* definido);

**4) Margin:** é a margem do elemento, a área fora da borda, sendo ele transparente.

**OBS:** o *outline* não está presente na *box model* justamente por ele tecnicamente não fazer parte da caixa, uma vez que ele irá se sobrepor a margem, isto é, caso eu possua uma margem de 20px e um contorno de 20px, este contorno irá passar por cima de toda a minha margem, fazendo com que visualmente não haja margem nenhuma no meu elemento.

**Altura e largura**

Assim como é possível definir a altura e largura de imagens, é possível definir estas propriedades para qualquer elemento. Como padrão, podemos utilizar qualquer tipo de medida para definir a altura e largura de um elemento.

**width:** define a largura do elemento;

**height:** define a altura do elemento.

h1 {

width: 1280px;

height: 720px;

}

Porém, caso usemos estas propriedades, uma vez que redimensonamos nosso navegador, o elemento continuará possuindo 1280x720px, e isso pode ser um problema para telas menores, ou mesmo para dispositivos mobile. Para resolver isso, há quatro propriedades essenciais para auxiliar com o design responsivo (página que responde corretamente para qualquer tamanho de tela):

**max-width:** define a largura máxima do elemento;

**max-height:** define a altura máxima do elemento;

**min-width:** define a largura mínima do elemento;

**min-height:** define a altura mínima do elemento.

h1 {

max-width: 1280px;

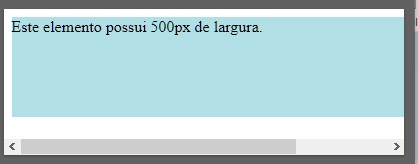
max-height: 720px;

}

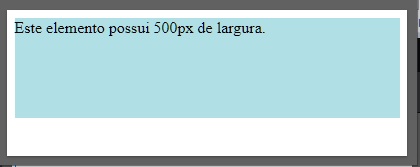
Neste caso, caso fosse redimensionado o navegador, ainda assim o elemento cabeira normalmente e funcionalmente na tela, sem o problema de parte dele ficar escondido na lateral, fazendo com que o usuário seja obrigado a utilizar a barra de scroll inferior para poder vê-lo novamente.

Ainda, a ideia com o “min” é a mesma, porém de forma invertida, ou seja, se o meu documento tem a largura mínima de 500px, quando eu abrir a página em uma tela de 1920px de largura, o elemento irá ocupar os 1920px da tela, e na medida que ele for sendo redimensionado, quando chegar em 500px, ele irá parar de diminuir de largura e ficar fixo em 500px, criando uma barra de scroll inferior dos 499px para baixo.

Com isso, temos que:



Com a propriedade “width” com valor de 500px em uma tela menor que 500px (scroll bar inferior aparecendo).



Com a propriedade “max-width” com valor de 500px em uma tela menor que 500px (scroll bar inferior não aparecendo).

**OBS¹:** o valor da propriedade “max-width” sobrescreve o valor da propriedade “width”, e o mesmo se aplica ao “max-height” e “height”.

**Formatação de texto**

Sobre a formatação do texto no CSS, podemos definir diversas propriedades, sendo elas relacionadas a cor, alinhamento, decoração, transformação, espaçamento e fonte.

**1) Propriedades de cor**

**1.1) color:** utilizada para mudar a cor do texto, compreende quaisquer valores de cor;

**1.2: background-color:** utilizada para mudar a cor do fundo do texto, compreende quaisquer valores de cor.

**2) Propriedades de alinhamento**

**2.1) text-align:** utilizada para definir o alinhamento horizontal do texto segundo onde ele está presente. Compreende os valores “middle”, “left”, “right” e “justify” (justificar texto);

**2.2) vertical-align:** utilizada para definir o alinhamento vertical do texto segundo onde ele está presente. Compreende os valores “top”, “middle”, e “bottom”;

**2.3) direction e unicode-bidi:** utilizada para colocar o texto sendo escrito da direita para a esquerda e com as letras invertidas. É provavelmente utilizado para escritas que precisem começar da direita, como o Árabe. Para usar essas propriedades, o direction deve ter o valor “rtl” e o Unicode-bidi o valor “bidi-override”.

**3) Propriedades de decoração**

**3.1) text-decoration:** utilizada para colocar decorações no texto. Compreende os valores “none”, “overline”, “line-through” e “underline”;

**3.2) text-shadow:** utilizada para adicionar uma sombra atrás do texto. Para definir seus valores, é necessário colocar “text-shadow: tamanho tamanho cor”.

**4) Porpriedades de transformação**

**4.1) text-transform:** utilizada para transformar as letras do texto, podendo deixa-las em maiúsculas, minúsculas ou de forma capitalizada (cada palavra começa com uma letra maiúscula). Compreende os valores “uppercase”, “lowercase” e “capitalize”.

**5) Propriedades de espaçamento**

**5.1) text-indent:** utilizada para definir um recuo no início do texto. Compreende valores de tamanho.

**5.2) letter-spacing:** utilizada para definir o espaço entre as letras. Compreende valores de tamanho.

**5.3) line-height:** utilizada para definir o espaço entre as linhas. Compreende valores de tamanho.

**5.4) word-spacing:** utilizada para definir o espaço entre as letras. Compreende valores de tamanho.

**5.5) white-space:** utilizada para desabilitar a quebra automática de texto com o valor “nowrap”

**Fontes de texto**

No CSS há cinco tipos diferentes de fontes genéricas:

**1) Serif:** são fontes que possuem pequenos contornos nas bordas de cada letra, trazendo um visual mais elegante e formal. Exemplo: Times New Roman;

**2) Sans-Serif:** são fontes limpas, que não possuem contornos em suas bordas, trazendo um visual mais simples e minimalista. Exemplo: Verdana;

**3) Monospace:** são fontes que todas as letras possuem exatamente a mesma largura, trazendo um visual mais mecânico. Exemplo: Consolas;

**4) Cursive:** são fontes que imitam a escrita à mão. Exemplo: Comic Sans MS;

**5) Fantasy:** são fontes que trazem visuais mais decorativos. Exemplo: Cooper Black.

Podemos utilizar qualquer fonte para estilizar nossa página, e para isso o CSS fornece diversas propriedades:

**1) font-family:** define a fonte que será usada. Aqui deve ser introduzido o nome da fonte;

**2) font-size:** define o tamanho da fonte, aceitando todos os tipos de medidas;

**3) font-style:** define se a fonte será normal, itálica ou oblíqua (é a mesma coisa que itálico);

**4) font-variant:** define se a fonte será normal ou então será em “small-caps” (todas as letras permanecem do mesmo tamanho, porém ficam em CAPS LOCK) ;

**5) font-weight:** define se a fonte será normal ou em *bold* (negrito). Essa propriedade aceita outras valores, mas dificilmente são utilizados, como “bolder”, “lighter” e até valores de 100 a 900, mudando apenas a centena;

**6) font:** é a *shorthand property* para modificar a fonte, para especificar todas as propriedades individualmente em uma única linha.

h1 {

font-family: ‘Arial;

font-size: 40px;

font-style: italic;

font-variant: small-caps;

font-weight: bold;

}

h2 {

font: italic small-caps bold 40px Arial;

}

**OBS:** As melhores fontes para serem utilizadas em uma page são: Arial, Verdana, Helvetica, Tahoma, Trebuchet MS, Times New Roman, Georgia, Garamond, Courier New, Brush Script MT.

**Ícones**

Para colocar ícones já pré-feitos no CSS é possível utilizar *libraries* (bibliotecas) de ícones. As três *libraries* mais utilizadas são: Font Awesome, Bootstrap glyphicons e Google Icons.

Para utilizar os ícones é usado qualquer elemento inline (como <i> ou <span>), e todos os ícones podemos ser customizados pelo CSS (cor, tamanho, sombra etc).

**1) Font Awesome:** para utilizar os ícones desta *library* é preciso se cadastrar no site para adquirir um link que será colocado dentro da tag head, como por exemplo:

<script src="https://kit.fontawesome.com/yourcode.js" crossorigin="anonymous"></script>

Após isso, basta adicionar os elementos definindo sua classe para o nome do ícone desejado:

Código do .HTML:

<!DOCTYPE html>

<html lang=”pt-BR”>

<head>

<script src="https://kit.fontawesome.com/yourcode.js" crossorigin="anonymous"></script>

</head>

<body>

<i class=”fas fa-cloud”></i> /\* ÍCONE DE NUVEM \*/

<i class=”fas fa-heart”></i> /\* ÍCONE DE CORAÇÃO\*/

</body>

</html>

**2) Bootstrap Glyphicon:** para utilizar os ícones desta *library* é preciso apenas colocar um link dentro da tag head:

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

Após isso, basta adicionar os elementos definindo sua classe para o nome do ícone desejado:

Código do .HTML:

<!DOCTYPE html>

<html lang=”pt-BR”>

<head>

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

</head>

<body>

<i class=”glyphicon glyphicon-cloud”></i> /\* ÍCONE DE NUVEM \*/

<i class=”glyphicon glyphicon-thumbs-up”></i> /\* ÍCONE DE JOINHA\*/

</body>

</html>

**3) Google Icons:** para utilizar os ícones desta *library* é preciso apenas colocar um link dentro da tag head:

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

Após isso, basta adicionar os elementos definindo sua classe para definir que é um ícone, e seu conteúdo que deve estar o nome do ícone desejado:

Código do .HTML:

<!DOCTYPE html>

<html lang=”pt-BR”>

<head>

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

</head>

<body>

<i class=”material-icons”>cloud</i> /\* ÍCONE DE NUVEM \*/

<i class=” material-icons”>computer</i> /\* ÍCONE DE COMPUTADOR\*/

</body>

</html>

**OBS¹:** o link para o site Font Awesome é: https://fontawesome.com/

**OBS²:** para ver todos os ícones disponíveis de forma organizada, visite:

https://www.w3schools.com/icons/default.asp

**Formatação de links**

Os links podem ser estilizados com todos as propriedades vistas na formatação de texto e fontes, porém, é possível estiliza-los mais ainda segundo o seu *state* (estado). Para fazer isso, é necessário utilizar dois pontos “:” e então definir o que ocorrerá caso tal estado esteja ativo (obviamente utilizando o elemento <a>). Os *states* que podem ser utilizados são:

**1) link:** definirá como o link será estilizado por padrão;

**2) visited:** definirá como o link será estilizado após ser visitado;

**3) hover:** definirá como o link será estilizado quando a seta do mouse passar por cima do link;

**4) active:** definirá como o link será estilizado quando ele for clicado.

Código do .CSS:

a:link {

font-size: 24px;

color: blue;

}

a:visited {

color: green;

font-family: “Times New Roman”;

}

a:hover {

background: yellow;

text-decoration: underline;

}

a:active {

letter-spacing: 20px;

}

**Estilização de listas**

Há dois tipos de listas no HTML: *unordered* (listas marcadas com símbolos/imagens) e *ordered* (listas marcadas com números ou letras)*.* Com o CSS é possível: colocar diferentes marcadores para listas ordenadas e não ordenadas, colocar uma imagem como marcador, adicionar um background para cada lista e itens da lista.

**1) list-style-type:** define qual será o marcador da lista. Aceita os valores “none” para quaisquer listas, “circle”, “disc” e “square” para listas não ordenadas e “decimal”, “decimal-leading-zero”, “upper-roman”, “lower-roman”, “upper-alpha” e “lower-alpha” para listas ordenadas (há outros valores, porém dificilmente serão utilizados alguma vez, como “hiragana” ou “armenian”);

**2) list-style-image:** define uma imagem como marcador da lista;

**3) list-style-position:** define se os marcadores ficarão dentro ou fora dos itens da lista, isto é, caso os itens estejam para fora dos itens da lista, logo eles ficarão fora de uma “border”. Aceita os valores “inside” e “outside”;

**4) list-style:** é a *shothand property* para a estilização da lista.

ul.a {

list-style-type: square;

}

ul.b {

list-style-image: url('minhafoto.jpg');

}

ol.a {

list-style-position: outside;

}

ol.b {

list-style: upper-roman inside url('minhafoto.jpg');

}

**OBS¹:** as listas já vem com margin e padding definidos, portanto, para remove-los basta adicionar para a lista desejada as propriedades “margin:0px; padding: 0px;” no CSS. Obviamenta também é necessária a “list-style-type” definida como “none” para remover até mesmo os marcadores.

**OBS²:** obviamente é possível personalizar as listas e seus itens com diversas outras propriedades (de texto, fonte, cor etc), porém, as apresentadas aqui são especificamente para listas. O CSS oferece uma grande variedade para estilização, portanto, é sempre necessário manter em mente todas as propriedades disponíveis para coisas consideradas “gerais”.

**Estilização de tabelas**

Para personalizar tabelas podemos utilizar diversas propriedades a serem aplicadas tanto na tabela, quanto nas colunas ou nas linhas.

**Adicionar umas borda da tabela:** adicionar a propriedade “border”;

**Remover as “bordas duplas”:** utilizar a propriedade “border-collapse” com o valor “colapse”. Isto fara com que as linhas se colapsem, removendo o efeito de “bordas duplas”.;

**Aumentar o tamanho total da tabela, dos heads ou itens:** utilizar a propriedade “width” ou “height” para as tags de tabela (<table>, <tr>, <th>, <td>);

**Alinhar heads ou itens horizontalmente:** utilizar a propriedade “text-align” e definir o valor desejado;

**Alinhar heads ou itens verticalmente:** utilizar a propriedade “vertical-align” e definir o valor desejado (a tabela deve conter espaço suficiente para o item poder realmente se mover);

**Alinhar a tabela ao centro:** utilizar a propriedade “margin” com os valores “0px auto” para a tag “table”;

**Alinhar a tabela na direita:** utilizar a propriedade “margin-right” com “0px” e “margin-left” com “auto”;

**Aumentar o tamanho de cada linha:** utilizar a propriedade “padding” com o valor desejado;

**Colocar uma borda apenas na parte inferior de cada linha:** utilizar a propriedade “border-bottom” para cada head e item;

**Mudar a certas propriedades visuais de cada linha ao passar o mouse por cima:** utilizar o seletor “:hover” no <tr>, e então definir as declarações desejadas;

**Propriedade Display**

A propriedade “display” especifica se ou como um elemento é mostrado.

Todo elemento de HTML possui um valor padrão de display, dependendo do elemento que é, sendo os mais comuns o “block” e o “inline”.

**1) block:** um elemento em bloco sempre utiliza todo o espaço disponível de um lado ao outro e adiciona uma quebra de linha automática. Alguns elementos com display em block são: <div>, <h1>, <p>, <form>, <header>, <footer> e <section>;

**2) inline:** um elemento em linha não utiliza todo o espaço disponível de um lado ao outro e não também não adiciona nenhuma quebra de linha, isto é, eles utilizam apenas o espaço necessário para o seu conteúdo, e assim, portanto, outros elementos colocados logo após ele estarão do seu lado, e não da linha de baixo; Alguns elementos com display em inline são: <span>, <a> e <img>.

**3) none:** um elemento “none” (nada) é aquele que não ocupa espaço na tela e que é invisível;

**4) visibility:hidden:** um elemento “hidden” (escondido) é aquele que ocupa espaço na tela e que é invisível;

**OBS:** o “none“ e “hidden” são comumente usados em JavaScript, para esconder e mostrar elementos sem a necessidade de os excluir e recriar.

**Propriedade Position**

A propriedade *position* (posição) é utilizada para definir o método de posicionamento de um elemento. Ainda, é possível utilizar quatro propriedades para personalizar a propriedade de posicionamento, que pode ser resumido como TBLR (“top”, “bottom”, “left” e “right”). Há cinco valores que podem ser utilizados na propriedade *position*:

**1) static:** é um elemento que não é posicionado de nenhum forma especial, ele simplesmente segue o fluxo do layout da page.

Este é o valor padrão dos elementos HTML. Elementos com posição estática não aceitam as propriedades “top”, “bottom”, “left” e “right”;

**2) relative:** é um elemento posicionado de forma relativa da sua posição padrão. Para utiliza-lo, ainda é necessário fazer o uso das propriedades de TBLR, pois caso não elas não sejam utilizadas, o posicionamento relativo não fará nenhuma diferença.

Ainda, a posição relativa não faz com que outros conteúdos preencham espaços que forem eventualmente deixados pelo elemento com tal posição, isto é, se ele ocupa o topo de uma page, o espaço dele continuará lá, pouco importando se graças ao TBLR ele se encontra na parte mais inferior da page.

E por fim, se houver um <span> com posição relativa e outro não, e este primeiro <span> ter sua propriedade “left” em 30px, este span irá ficar literalmente por cima do outro <span>, portanto, é como se o espaço ocupado pelo elemento continuasse no mesmo local, mas sua parte visual fosse para outro lugar. Também, como dito no parágrafo anterior, o espaço ocupado pelo span com posição relativo continuará lá, e portanto, deixará um vão em branco onde seria sua posição original;

**3) fixed:** é um elemento que terá sua posição fixa na viewport, isto é, mesmo que a página possua uma barra de scroll longa, independetemente onde o usuário estiver, este elemento estará fixo no mesmo local.

Ele se sobrepõe visualmente sobre outros elementos caso suas propriedades de TBLR façam com que ele esteja em cima de outros elementos, e também ele não deixa espaços em branco na page, portanto, se difere da posição relativa neste aspecto, isto é, se um elemento com posição fixa estiver originalmente no topo, mas sua posição for modificada através do TBLR, logo o espaço que ele ocupava na parte do topo não será mais considerada, e outros elementos preencherão este vão;

**4) absolute:** é um elemento que terá sua posição relativa ao seu elemento *parent* (pai), isto é, se a viewport possui 1920x1080px, e há uma div1 de 500x500px (com qualquer posição) com outra div2 de 100x100px (com posição absoluta), a div2 será posicionada relativa ao tamanho da div1, e não ao tamanho da viewport, portanto, caso seja colocado como declaração “right:0px”, a div2 ficará na borda da div1 de 500x500px, e não na borda da viewport de 1920x1080px.

Dessa, é possível perceber a diferença com a posição fixada, a qual o elemento fica fixo na viewport, e não no seu elemento pai

**5) sticky:** é um elemento que terá sua posição definida com base no scroll do mouse do usuário.

A posição *sticky* (grudenta) alterna entre a posição *relative* e *fixed* com base no scroll do mouse do usuário, isto é, o elemento possui primeiramente uma posição relativa, com exatamente o mesmo comportamento do valor *relative*, porém, quando o elemento for sair da tela do usuário ele entrará com a posição em valor *fixed*, e dessa forma, ele sempre estará visível na tela a partir de determinada rolação de scroll da page, e ainda, sua posição por padrão ficará no topo, mas com as propriedades TBLR ele poderá ficar mais abaixo, mais a direita etc.

**OBS¹:** caso o elemento com posição absoluta não possua nenhum elemento pai, isto é, seu “elemento” pai seja a própria viewport, ele será posicionado com base na viewport;

**OBS²:** positioned elements (elementos posicionados) são elementos com o valor da propriedade *position* diferente de *static*;

**OBS3:** para resolver o problema de um elemento posicionado estar sobrepondo visualmente outro elemento, basta utilizar a propriedade “z-index”, que basicamente define qual elemento deve ficar mais para trás, qual deve ficar mais na frente, qual deve ficar ainda mais nada frente (como se fosse um sistema de camadas do Photoshop). O “z-index” compreende valores numéricos, sendo que quanto maior o valor, mais na frente o elemento estará, e quando menor o valor, mais atrás ele estará.

**Propriedade Overflow**