**APRENDENDO HTML**

<!DOCTYPE>

A declaração <!DOCTYPE> não é uma tag HTML, ela nada mais é que uma instrução para o navegador web, sobre qual versão do HTML a página está sendo escrita.

TAGS HTML:

As tags HTML servem como etiquetas em nossos códigos HTML. Essas etiquetas servem para informar ao navegador que o que está entre as tags deverá ser interpretado de uma maneira diferente.

* Definição das tags em HTML:

As tags vêm entre os símbolos de < >.

A grande maioria delas existem em pares, indicando aonde deve iniciar e terminar um tipo de marcação. A tag de início é somente o comando entre < >, já a tag de termino, possuí o símbolo / após o <. Veja:

Tag de início: <nome\_do\_comando>

Tag de termino: </nome\_do\_comando>

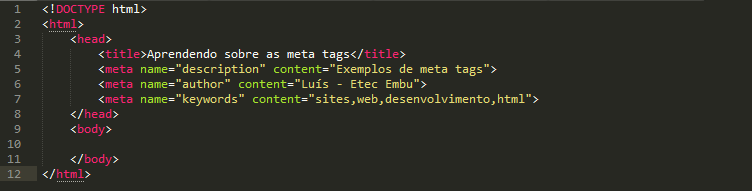
META-TAGS:

Meta-tags são linhas de comando HTML que descrevem o conteúdo do seu site para os buscadores. É nelas que você deve configurar ações que ajudaram os usuários e sites de buscas a te encontrar na internet, além de outras definições.

As meta-tags devem ser incluídas dentro da tag <head>. Veja alguns exemplos de meta-tags:

* *Description*: Esta tag tem a função de exibir a descrição da sua página. É o conteúdo inserido nela que irá aparecer embaixo do título dos buscadores.
* *Author:* É responsável por dizer quem é o autor do site.
* *Keywords*: Esta tag deve incluir uma quantidade de palavras que se referem ao conteúdo da página. Elas auxiliam o seu site na web.

EXEMPLO:

**

ORGANIZANDO O CODIGO:

Dois princípios são utilizados para melhorar a estrutura dos códigos HTML, aninhamento e indentação.

-Tags aninhadas, são aquelas que estão alocadas uma dentro das outras, mas nunca de forma aleatória.

Ex: *<casa>*

*<quarto>*

*<banheiro>*

*Estamos no banheiro*

*</banheiro>*

*</quarto>*

*</casa>*

Repare, na estrutura de aninhamento a primeira tag que eu abri, sempre será a última a se fechar. Seguindo uma sequência.

- Indentação, nada mais é do que organizar seu código, fazer com que ele fique esteticamente melhor. Mas, não é apenas questão estética, pois um código indentado facilita encontrar determinada infomação.

As tags correspondentes (abertura/fechamento) devem ficar na mesma linha VERTICAL e as tags aninhadas devem ficar a frente.

Ex: *<casa>*

*<quarto>*

*Estamos no quarto!*

*<banheiro>*

*Agora, estamos no banheiro do quarto.*

*</banheiro>*

*</quarto>*

*<cozinha>*

*Agora estamos na sozinha.*

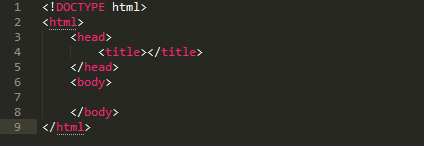
*</cozinha>*

*</casa>*

*Obs.:* Lembrem-se, ao criar o arquivo HTML, vocês devem atentar se ao salvar o arquivo o mesmo já esteja como extensão .html, caso não esteja é imprescindível que vocês coloquem o nome\_do\_arquivo.html para que seu navegador interprete.

UTILIZANDO PRIMEIROS COMANDOS HTML:

* Estrutura base:



Como já foi dito, o comando <!DOCTYPE> para expressar qual a versão do HTML está sendo desenvolvido.

<html> Envolve todo um documento html. </html>

<head> Envolve o cabeçalho de um documento html. </head>

<title> O título do documento. </title>

<body> Envolve o corpo (texto e tags) do documento html. </body>

Estas são as tags da estrutura base do html, a partir deste ponto, vamos moldando e adicionando novas tags para poder montar a página.

***DICA:*** Organize seus projetos em pastas onde todos os arquivos pertencentes a ele estejam alocados no mesmo local. Desta forma você centraliza as informações e evita que ao alterar o carregar seu arquivo para um outro dispositivo o mesmo perca informações.

Você pode organizar o seu projeto da seguinte forma: Crie uma pasta superior, informando de qual projeto se trata, dentro desta pasta superior crie subpastas, uma para cada tipo de arquivo que você possa trabalhar. Ex: Imagens, CSS, JavaScript, Videos, Audios e etc. Desta forma você pode organizar seus arquivos a partir de suas categorias fazendo com que o mesmo não se perca na troca de dispositivos e facilitando ainda mais a codificação na hora de relaciona-los. Os arquivos HTML podem ficar fora das subpastas, apenas dentro da pasta mestre.

Nomeie as os arquivos HTML com o devido nome de acordo com o que a mesma trata. Porem a primeira página, ou seja, sua página home deve ser nomeada como index. Veja o exemplo abaixo de arvore de diretórios.

******Desta forma, quando você for referenciar os arquivos que for utilizar em seu projeto, basta apenas colocar o nome da pasta referente a ele no caminho/nome do arquivo e a extensão.

**ALTERAÇÃO DE CORES**

Existem mais de uma forma de mudar as cores de letras, bordas e fundos em HTML, uma delas é colocando o nome da cor em inglês, como vimos na aula passada. A segunda maneira é através da sua representação em hexadecimal, podendo ter uma gama maior de cores. Uma outra maneira é através do código RGB de cada cor. Podendo alcançar mais de 16 milhões de cores.

Color: blue;

Color: #F0B9E2;

Color: rgb (200, 160, 90);

Uma outra funcionalidade que o HTML 5 trouxe é a possibilidade de usar o RGBA. Com o RGBA, você pode colocar um valor entre 0 e 1 para delimitar uma opacidade para o objeto em si.

Color: rgba (50, 240, 120, 0.3)

COMO ALTERAR CORES DE FUNDO NA LETRA:

Nós podemos colocar uma cor no fundo das nossas letras, chamado background. Editamos no nosso style css do objeto.

*Background-color: red;*

Podemos alterar a cor dos textos usando o seguinte comando em CSS:

*P {*

*Color: Orange;*

*}*

Vale lembrar que da forma acima, você está configurando para todos os objetos “p”. Caso queira formatar pequenos trechos de uma frase, pode utilizar a formatação direto no HTML.

*<font color=”blue”> TEXTO </font>*

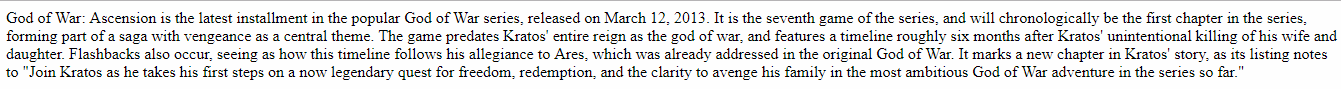
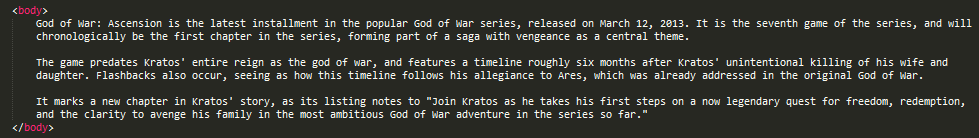
Podemos alterar a cor de fundo das nossas páginas, tanto na forma HTML como na CSS:

EM HTML: *<body bgcolor = “purple”>*

*EM CSS: Body{*

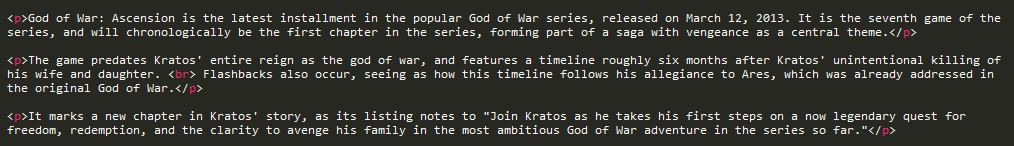
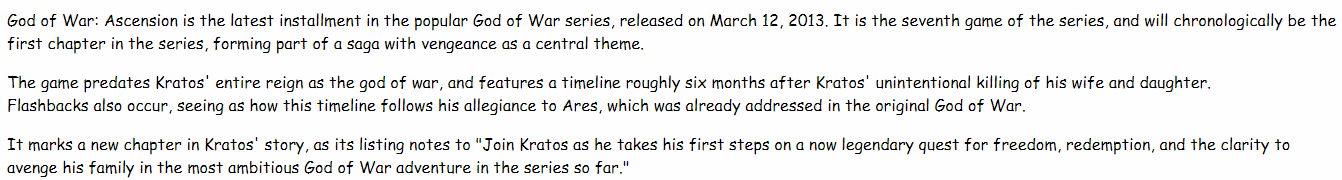
*Background-color: black ou #05F488;*

FORMATANDO TEXTOS:

Os navegadores não conseguem interpretar formatações de textos como por exemplo quebras de linhas, parágrafos, centralizações e afins. Reparem abaixo, por mais que colocamos espações entre os parágrafos em nossos códigos, veja como fica tudo amontoado no resultado do navegador.

Temos então algumas tags para auxiliar na formatação e organização dos textos em nossas páginas.

<p> Cria parágrafos de acordo com a área delimitada, tudo o que estiver entre a tag p será um parágrafo.</p>

<br> A tag br é utilizada para a quebra de linha, caso você precise fazer uma quebra de linha, pode utiliza-la. Esta tag não tem fechamento, apenas a insira aonde desejar fazer a quebra.

Desta forma com a inserção das tags, o navegador já consegue nos exibir o texto de uma forma mais organizada. Podemos passar alguns parâmetros para a tag <p> e informar como queremos a distribuição do texto da seguinte forma.

<p align=”left”>TEXTO FICA AJUSTADO A ESQUERDA</p>

<p align=”center”>TEXTO FICA AJUSTADO NO CENTRO</p>

<p align=”right”>TEXTO FICA AJUSTADO A DIREITA</p>

<p align=”justify”>TEXTO FICA AJUSTADO COMO JUSTIFICADO</p>

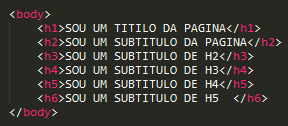
Podemos utilizar os comandos abaixo também para:

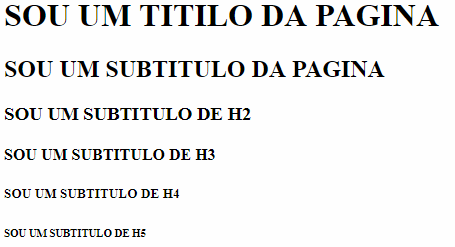
<strong>ITEM 1</strong> - **TEXTO EM NEGRITO**

<em>ITEM 2</em> - *TEXTO EM ITALICO*

<u>ITEM 3</u> - TEXTO SUBLINHADO

Temos também as tags de título chamadas de H, estas tags vão de H1 a H6 onde H1 é definido como título principal, H2 como subtítulo de H1, H3 como subtítulo de H2 que consequentemente é subtítulo de H1 e assim por diante. Veja abaixo como usar a tag e seu resultado.





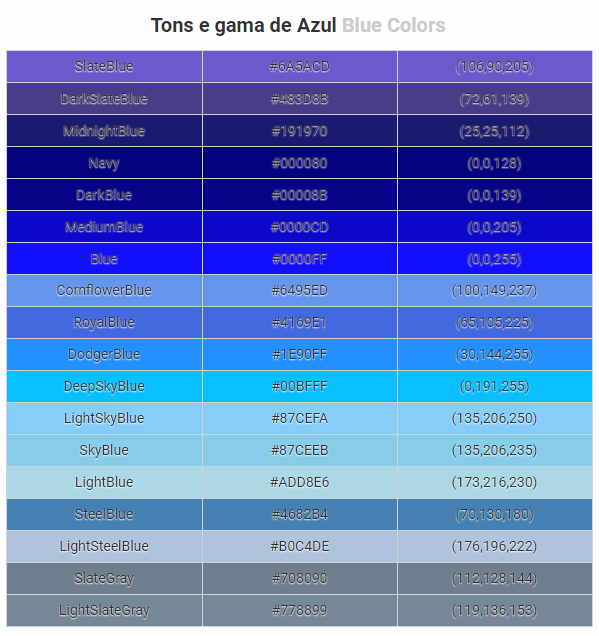
Podemos formatar e personalizar nossos textos, tanto para a tag H como para a tag P. Para utilizarmos em CSS e replicar as configurações padronizando o texto devemos criar uma tag <style> dentro da nossa tag <head> e configura-la ali. Posteriormente, vamos aprender a trabalhar com CSS e modificar as configurações desta tag Style para um arquivo único. Abaixo exemplo de como personalizarmos.

 Configuramos as características que seriam adotadas para tudo o que for <h1>, <h2> e <h3> trabalhando com os atributos de font-family escolhendo o tipo de fonte da letra, color especificando a cor da letra e para dar um efeito usamos o parâmetro text-shadow para aplicar uma sombra sob as letras.

O primeiro parâmetro passado no text-shadow é o deslocamento horizontal da sombra, o segundo parâmetro é o descolamento vertical o terceiro é o espalhamento dela.

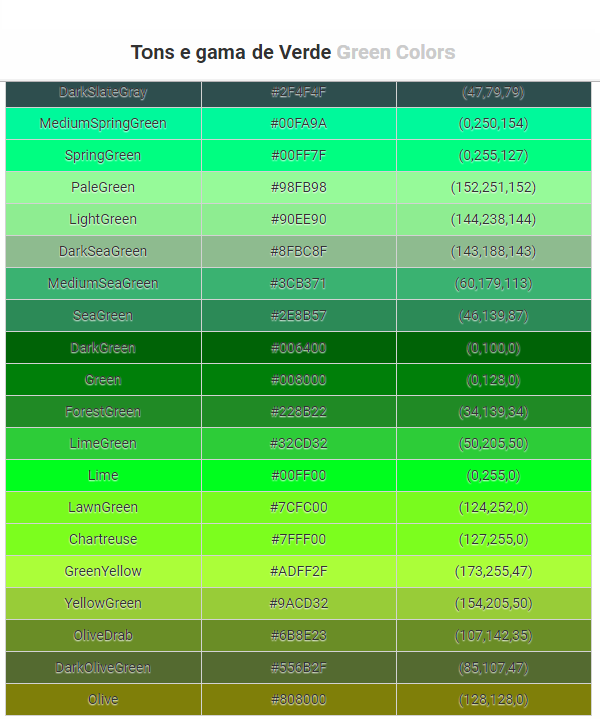
Notem que a tag <style> foi aberta dentro da tag <head> e após as configurações a mesma foi fechada.

Abaixo uma lista com algumas cores representadas com nome, código hexadecimal e código rgb.

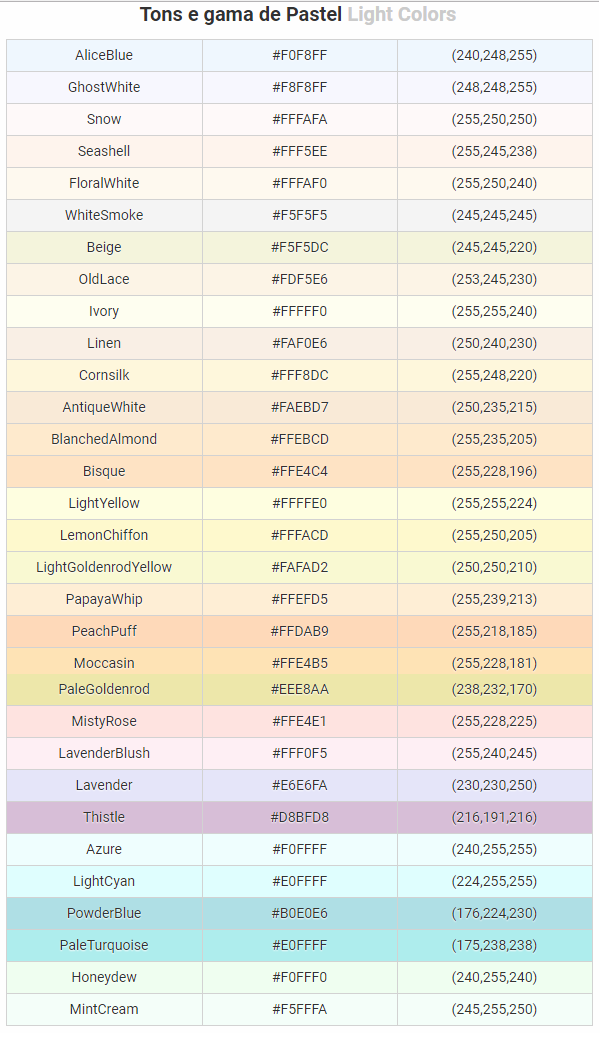










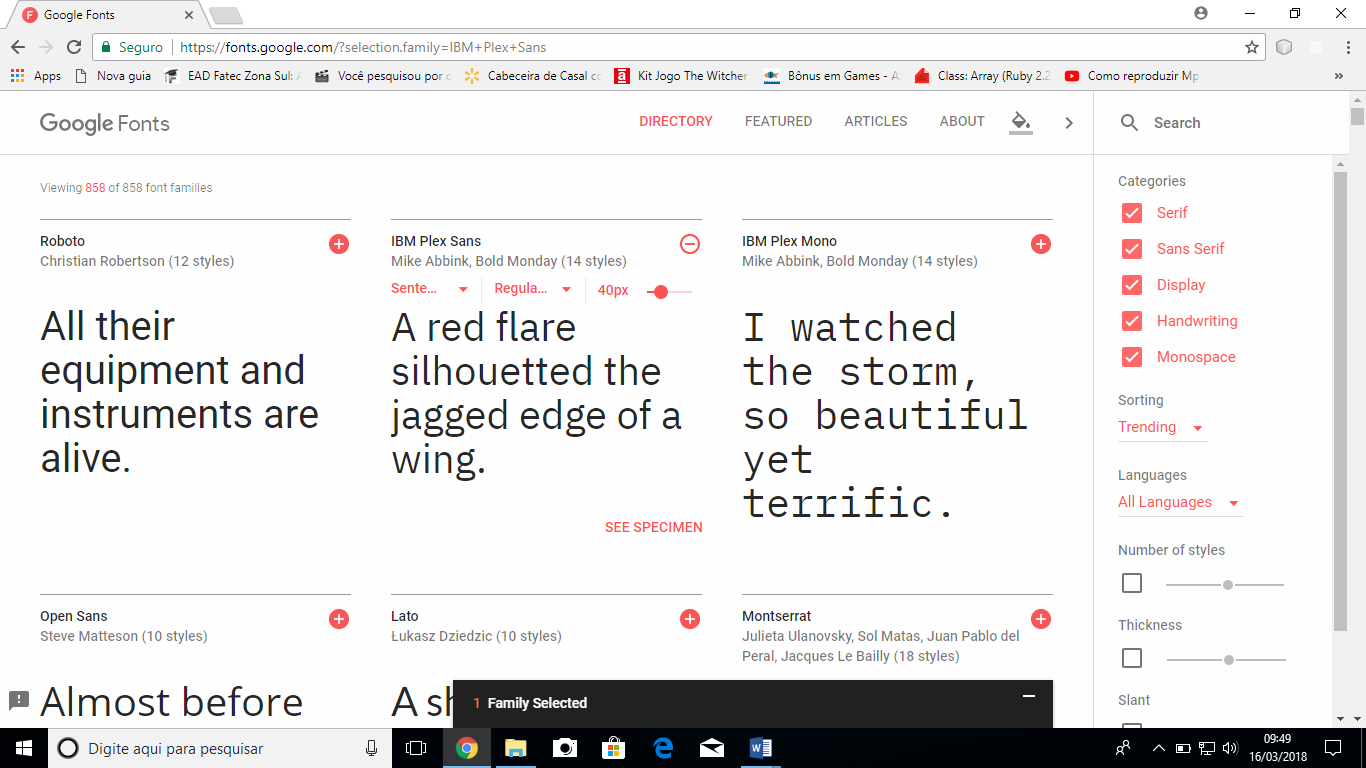


TRABALHANDO COM DIFERENTES FONTES:

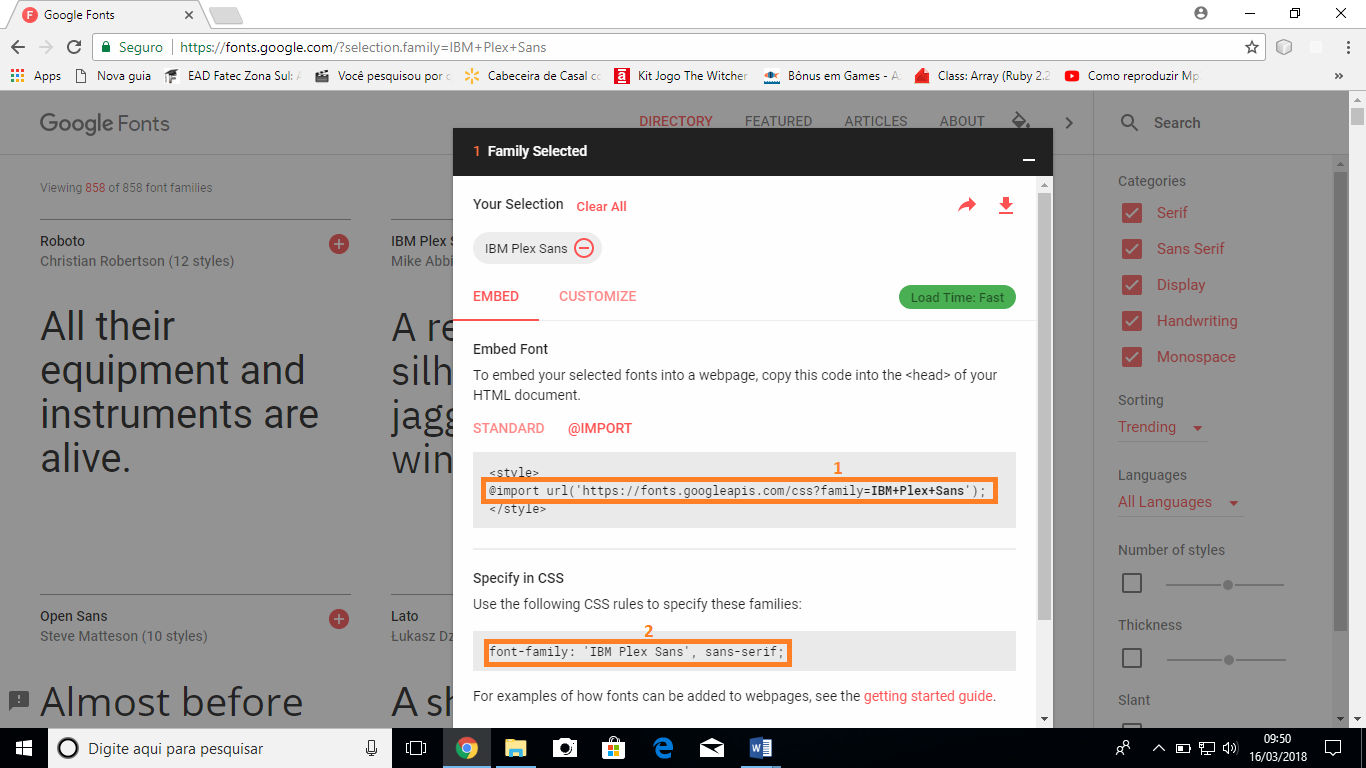
Outra mudança possível é na fonte, uma boa dica é para utilizarmos o *google.com/fonts* e utilizar fontes publicas diretamente de lá. Para utilizar, devemos copiar a biblioteca e depois referenciar no CSS.

Como fazer?

Acesse o site do *google.com/forms* selecione a fonte que desejar e clique no sinal de “+” em vermelho. Feito isso, aparecerá uma janela chamada “Family Selected” como na imagem abaixo.



Após aparecer a janela da fonte selecionada, clique nela e depois clique na parte “@import”.



Copie todo o comando do @import expresso na figura acima pela retângulo 1 e cole-o dentro do seu código CSS.

*@import url (“endereço’);*

Selecione também o comando representado no retângulo número 2 e cole-o dentro do objeto que deseja fazer a modificação.

*H1 {*

*Font-family: ‘nome’, cursive;*

*}*

No exemplo acima, foi definido que apenas os objetos de H1 receberam esta formatação. Caso queira expandir para todo o seu site, deverá colocar o comando acima dentro das configurações de body.

HTML e semântica

Devemos descrever bem o conteúdo para que leitores de tela e buscadores consigam processar melhor esse conteúdo e, assim, tornar a informação mais acessível aos nossos visitantes.

Em outras palavras, devemos tentar deixar o nosso código HTML semântico, usando, para isso, as tags mais adequadas a cada tipo de conteúdo.

Visualmente, não teremos nenhuma diferença, mas agora os leitores de tela podem oferecer um atalho que leve o usuário mais rapidamente ao menu de navegação do site e os buscadores podem oferecer esses links no resultado da busca. Um HTML semântico melhora a acessibilidade do site para todos.

Um HTML semântico é tão importante que, em sua última atualização, a linguagem HTML incluiu diversas novas tags cujo único propósito é permitir uma marcação mais semântica para nossas páginas! Além da tag <nav> que acabamos de usar, temos diversas outras:

*<main>: conteúdo principal da página*

*<header>: cabeçalho da página ou de uma região dela*

*<footer>: mesma ideia da tag <header> para o rodapé*

*<aside>: conteúdo auxiliar ao conteúdo principal, como links relacionados ao conteúdo*

*<article>: conteúdo que, por si só, já tem um sentido completo, como um post de um blog ou uma notícia*

*<section>: parte/seção de uma página ou texto*

Há ainda muitas outras, que estudaremos aos poucos. Por hora, podemos usar algumas das tags para deixar nosso código mais semântico. O texto da biografia, por exemplo, pode ser embrulhado numa tag <main>. O <nav> da lateral pode ser embrulhado numa tag <aside> junto com o título e os outros links que ainda precisamos colocar para essa parte. Podemos colocar um rodapé na página com uma mensagem de copyright com a tag <footer>.

TRABALHANDO COM LISTAS:

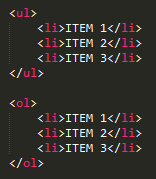
No HTML temos três estruturas para trabalhar com listas:

Listas não ordenadas - <UL>

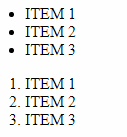
Listas ordenadas - <OL>

Listas de definições - <DL>

Abaixo exemplo da estrutura de listas:

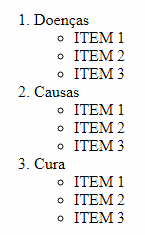
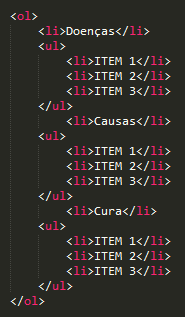


Primeiro definimos qual tipo de lista queremos trabalhar, se for uma lista não ordenada colocamos a tag <ul> e na sequencia dentro da tag <li> definimos quais os itens que pertencerão a esta lista. Funcionando na mesma estrutura da lista não ordenada, temos as listas ordenadas. Onde definimos sua estrutura e listamos os itens pertencentes a ela como visto no trecho de código acima e trazendo o resultado abaixo.



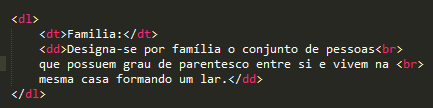
Vale observar que uma UL (lista não ordenada) nos mostra uma lista sem sequência numérica, já uma OL (lista ordenada) traz os resultados dentro de uma lista organizada sequencialmente.

Podemos utilizar uma lista dentro da outra, como no exemplo abaixo.

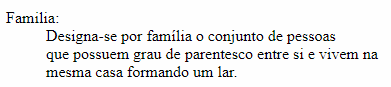


Notem que foi iniciado uma estrutura de lista ordenada e dentro da lista ordenada uma nova estrutura de lista não ordenada, fazendo com que chegássemos no resultado acima.

Além das listas UL e OL mostradas acima, temos também as listas DL. Estas listas trabalham com a mesma estrutura lógica das outras duas, no entanto usamos tags diferentes para trabalhar com elas. Veja:



Na estrutura acima definimos com a tag <dl> que vamos trabalhar com este tipo de lista, na tag <dt> especificamos o título ou a parte em destaque do termo proposto que teremos em nossa lista e com a tag <dd> podemos colocar a definição do termo. Exemplo de como ficará uma lista DL.



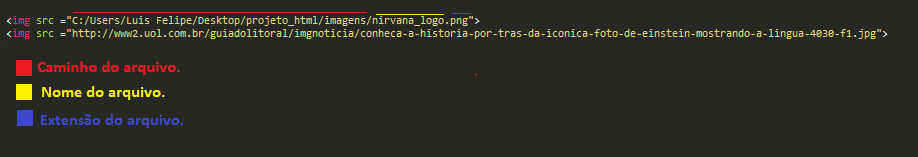
Notem que foi utilizado a tag <br> no código, esta tag como foi dito é utilizada para fazer a quebra de linhas dentro de uma página HTML. A tag <br> não tem sua tag de fechamento. Basta apenas coloca-la aonde desejar fazer a quebra de linha, sem a necessidade de fecha-lá.

TRABALHANDO COM IMAGENS:

Para inserirmos imagens dentro de nossas páginas HTML utilizamos o parâmetro:

<img src = ”caminho\_da\_imagem/nome\_arquivo.extensão”>

Podemos utilizar imagens salvas em nossos computadores ou pega-las diretamente da web. Veja abaixo os dois exemplos.



O primeiro exemplo é como podemos pegar uma imagem de nossos computadores e coloca-las na página HTML. Com os riscos acima são expressos de acordo com a legenda a estrutura, onde a parte que está em vermelho é o caminho do arquivo, em amarelo é o nome do arquivo que você escolherá e em azul a extensão deste arquivo.

O segundo exemplo mostra como colocar uma imagem diretamente da internet, selecionando a imagem desejada, clicando com o botão direito do mouse e copiando o endereço dela, basta que você cole o endereço dentro do campo do caminho.

Você pode exemplificar o caminho da sua imagem direto do computador não sendo necessário colocar todo o caminho como no exemplo acima. Para isso deverá colocar a seguinte estrutura.

<img src ="../projeto\_html/imagens/nirvana\_logo.png">

Desta forma, o **..** colocado no início do caminho abreviará toda a extensão anterior, tendo a necessidade apenas que você coloquei o nome da pasta aonde encontra o seu arquivo de foto e o nome da foto juntamente com a extensão.

Caso sua imagem esteja num tamanho diferente do que deseja, você pode alterar as propriedades de tamanho com os comandos *width* para alterar as medidas da largura da imagem e *height* para alterar as medidas da altura da imagem. Veja como fazer:



Após definir o caminho da imagem, adicione os parâmetros e modifique-os para ajustar a imagem no tamanho desejado. Caso queira modificar a imagem para esquerda ou direita utilize o parâmetro align=”left/right”, desta forma sua imagem ficará em um dos dois extremos.



Caso deseje que sua imagem fique alinhada no centro da página pode utilizar a tag <center> e definir o que deseja que fique centralizado.



Lembre-se sempre de abrir e fechar as tags que são necessárias, caso contrário, tudo aquilo que estiver dentro de <center> ficará centralizado.

No CSS você pode ajustar a posição da imagem em diferentes posições dentro da sua página fora as mencionadas acima, com o parâmetro:

Img {

Position: absolute;

Left: 350px;

Top: 45px;

}

Colocando imagem como background:

Voce pode definir uma imagem como fundo de duas maneiras, por HTML:

<body background= “caminho/imagem.jpg”>

Ou por CSS:

Body {

background-image: url(“caminho/imagem.jpg”);

}

No CSS você pode trabalhar com algumas configurações de parâmetros para o background:

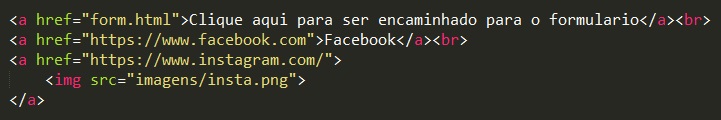


Caso você queira que esta imagem de background fique estática no fundo da sua página, deve adicionar no código da imagem acima o seguinte parâmetro:

background-attachment: fixed; Isto fará com que a imagem de fundo fique estática e apenas o conteúdo da sua pagina se mova com a barra de rolagem.

Inserindo Links:

Nós podemos inserir links internos e externos em nossos sites. Geralmente os links internos servem como ponte para alterarmos entre uma página e outra ou conteúdo e outro de nosso domínio, já os links externos nos levam a uma página de terceiros. Para isso utilizamos “a href”. Veja o exemplo:



Na primeira linha do exemplo, inserimos um link para uma outra pagina desenvolvida que faça parte do site. Ex: Home, quem somos, fale conosco...

No segundo exemplo, referenciamos um link externo, ao invés de apontar para uma página local, apontamos para o site do Facebook como exemplo.

Por fim, no terceiro podemos fazer um link de uma imagem com um site. No caso, utilizamos a tag de link e inserimos uma imagem dentro das tags de abertura e fechamento. <a> </a>.

Inserindo multimídia (áudios e vídeos):

Este recurso ainda vem sendo trabalhado e alguns navegadores não estão 100% compatível com todos os formatos disponíveis. Para que você possa inserir áudios e vídeos na sua página é necessário ter os arquivos salvos em um formato compatível. Geralmente mp3 e mp4.

Siga as instruções abaixo para inserir:



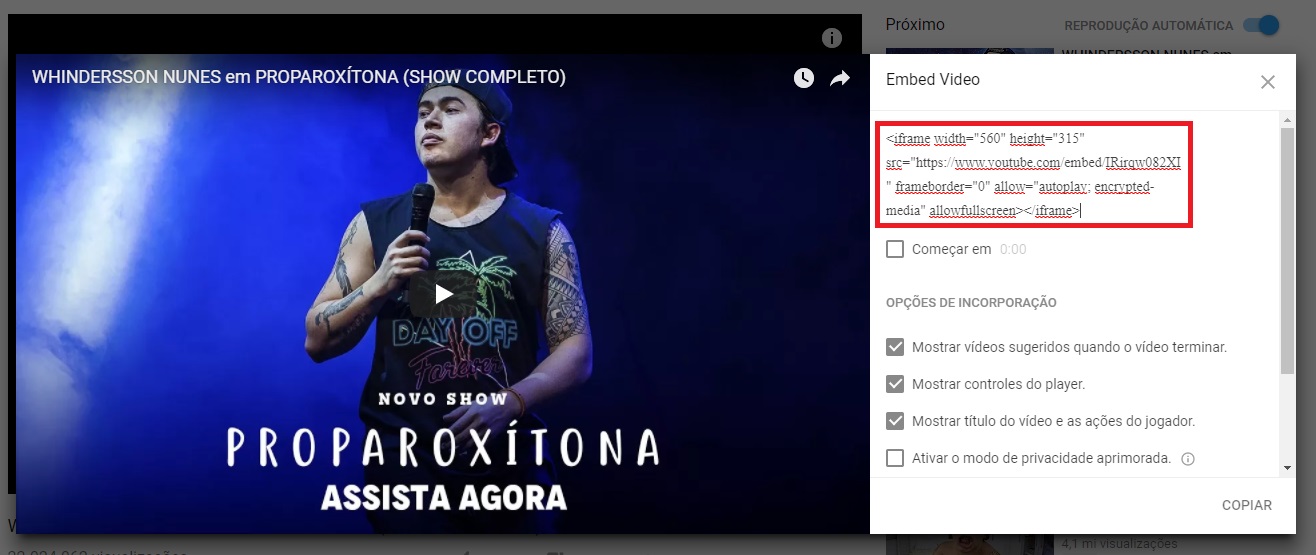
Observe que temos uma tag especifica para cada um dos tipos de arquivos.

<áudio> </áudio> - para sons e <vídeo> </vídeo> para vídeos.

Após você definir qual a tag utilizar de acordo com seu tipo de arquivo observe que você deverá passar um parâmetro chamado “controls”, caso este parâmetro não seja passado, o usuário do site não poderá dar o play nas mídias.

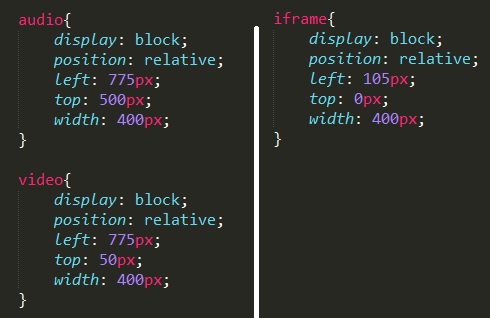
Na tag <source> você deve especificar o “*caminho/nome\_arquivo.extensão”* e o type pode manter como da foto, é uma propriedade padrão do HTML para informar qual o tipo do arquivo.

Caso você queirainserir um vídeo diretamente do youtube o próprio site do youtube disponibiliza uma configuração de ajuda. Vá na página do Youtube até o vídeo que deseja inserir, clique no botão “compartilhar” abaixo do vídeo e em seguida incorporar. Pegue a descrição disponibilizada e cole no seu código HTML.



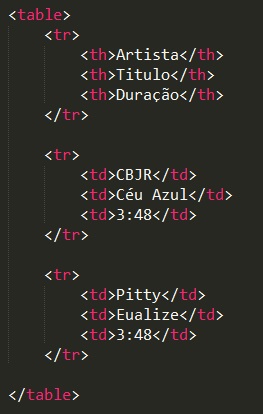


Podemos configurar estes arquivos com os seguintes parâmetros dentro no nosso CSS:



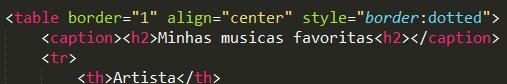
TRABALHANDO COM TABELAS:

Para utilizarmos tabelas precisamos inserir a tag <table> </table>. Esta tag define que vamos começar a trabalhar com uma tabela. Dentro da tag *table* utilizamos outras duas tag para a construção de uma tabela, a <tr> </tr> que são as **linhas** da tabela e a <td> </td> que são as **colunas**. Veja o exemplo:



Desta forma, criamos linha a linha nossas tabelas inserindo os dados.

Nós podemos personalizar as linhas da nossa tabela inserindo alguns parâmetros na sua criação:



Na borda é possível alterar sua espessura, a posição da tabela e o estilo da borda (solid, double, dotted, dashed).

Podemos alinhar os dados dentro de nossas colunas com o parâmetro *align=”left/center/right”*

Porém, fazer isso no HTML seria muito trabalhoso levando em consideração trabalhar com uma tabela grande e ter que fazer um por um, então podemos fazer no CSS e para isso basta:

th, td {

text-align: center;

}

Podemos alinhar também verticalmente com o parâmetro valign=”top/middle/bottom”

Temos um atributo chamado ***colspan*** e ***rowspan*** que permitem fazer a mesclagem das células. A ***colspan*** faz a mesclagem das células no sentido das colunas e a ***rowspan*** faz a mesclagem das células no sentido das linhas.

CONSTRUINDO UMA BASE SOLIDA:

Antes de começarmos a fazer os ajustes nos outros espaçamentos, repare em uma coisa: definimos os tamanhos das fontes dos títulos e subtítulos? Há espaço entre os parágrafos de texto e entre o título e as bordas da janela; fomos nós que colocamos esses espaços?

Desde a primeira aula, quando fizemos apenas o HTML, a página já tinha alguns estilos, **mesmo sem termos escrito nenhum CSS**.

Ou seja, **o navegador já dá um estilo padrão** para nossas páginas. E qual o problema disso? O problema é que nem sempre queremos esse estilo padrão. E mais: como é um estilo *do navegador*, cada navegador pode fazer o seu próprio estilo, como vimos anteriormente com as fontes.

Nosso site precisa funcionar bem em todos os navegadores, então usar como base para o nosso CSS o estilo padrão do navegador não é uma boa ideia.

Porém, nada impede de **sobrescrevermos** o estilo padrão do navegador, como já fizemos com as fontes! No caso dos espaçamentos, vimos que podemos mexer no espaço entre eles com a propriedade margin e no espaço dentro deles com a propriedade padding. Então, se quisermos tirar o espaço que o navegador coloca entre nossos parágrafos, podemos escrever no nosso CSS:

p {

margin: 0;

}

Podemos, também, querer tirar os tamanhos de fonte que o navegador coloca para os títulos e subtítulos:

h1, h2 {

font-size: 100%; /\* 100% = tamanho original da fonte do navegador \*/

}

Esses são apenas alguns exemplos. Na prática, precisamos sobrescrever várias regras que o navegador coloca por padrão, mesmo nos navegadores mais modernos, o que dá um bom trabalho. Para poder reaproveitar esse trabalho entre os nossos projetos, podemos até colocar essa sobrescrita num arquivo CSS separado, só para ele. Felizmente, vários desenvolvedores já tiveram essa ideia e disponibilizaram seus arquivos para que pudéssemos usar nos nossos próprios projetos.

Esse tipo de arquivo até ganhou um nome: reset. Atualmente, existem alguns disponíveis, com algumas diferenças entre eles. Alguns dos mais populares são:

Eric Meyer: arquivo bem pequeno; deixa todos os elementos com mesma aparência e tamanho de fonte; usa a fonte padrão do navegador

Normalize: arquivo um pouco maior, mas com ajustes mais finos; já define fontes e tamanhos padrão para alguns elementos; corrige diversas inconsistências entre navegadores

YUI (Yahoo!): arquivo bem pequeno, com um efeito bem parecido com o do Eric Meyer, mas com alguns ajustes mais específicos

No nosso projeto, vamos usar o reset do Eric Meyer, mas você pode usar qual você achar melhor. O importante é usar um para ter uma base sólida para desenvolver seu layout, independente de navegador.

Como o reset é só mais um arquivo CSS no projeto, para usá-lo, basta baixar o arquivo e importá-lo como já fizemos com o nosso próprio CSS:

<link rel="stylesheet" href="reset.css">

Podemos, sim, ter vários arquivos CSS para uma única página HTML. Mas onde colocamos essa tag? No <head>? Antes ou depois do nosso CSS? Veja só: no reset, teremos uma regra do tipo:

h1, h2, h3, h4, h5, h6 {

font-size: 100%;

}

Já no nosso arquivo, vamos querer fazer algo do tipo

h1 {

font-size: 30px;

}

Qual dessas duas regras o navegador deve usar? A ordem em que colocamos as tags <link> é importante. O navegador lê as regras sequencialmente e, se houver conflito entre regras, pega a última declaração. Ou seja, no exemplo acima, o navegador vai usar o tamanho de fonte 30px.

Então, se queremos usar o reset como base para o nosso CSS, devemos importá-lo antesdo nosso CSS na página:

<head>

<link rel="stylesheet" href="reset.css">

<link rel="stylesheet" href="bio.css">

</head>

Um pouco mais sobre conflitos no CSS

Acabamos de ver que, quando duas declarações conflitam no CSS, o navegador usa aquela que for declarada por último. Agora, o que será que acontece na seguinte situação?

h1 {

font-size: 100%;

color: red;

}

h1 {

font-size: 30px;

font-weight: bold;

}

Para entender como o navegador entende as declarações acima, precisamos entender que o conflito acontece no elemento onde o estilo vai ser aplicado. No exemplo acima, o estilo vai ser aplicado nos <h1>. Então o navegador vai fazer o seguinte:

Aplicar o font-size: 100%

Aplicar o color: red

Aplicar o font-size: 30px, sobrescrevendo o font-size: 100% anterior

Aplicar o font-weight: bold

Ou seja, o navegador vai simplesmente aplicando todas as regras que ele encontrar para um elemento. Se ele já mexeu numa propriedade (como o font-size no exemplo), não tem problema: ele mexe de novo.

A figura muda um pouco quando somos mais específicos na hora de selecionar um elemento. Imagine que agora temos o seguinte cenário: no HTML temos

<main>

<h1>Título principal</h1>

</main>

E, no CSS, temos

main h1 {

font-size: 30px;

color: red;

}

h1 {

font-size: 20px;

font-weight: bold;

}

Quando selecionamos elementos por hierarquia, estamos sendo mais específicos do que quando selecionamos apenas pela tag. Nesse caso, o navegador começa a aplicação dos estilos partindo das regras declaradas em seletores menos específicos. Ou seja:

Aplica o font-size: 20px do seletor h1

Aplica o font-weight: bold do seletor h1

Aplica o font-size: 30px do seletor main h1, sobrescrevendo o font-size: 20px

Aplica o color: red do seletor main h1

Então não apenas a ordem no CSS importa, mas também a especificidade dos seletores que usamos. Existem várias regras para definir a especificidade de um seletor e não vale a pena entrar em detalhes aqui. O importante é saber que, quando selecionamos um elemento, ser muito ou pouco específico pode atrapalhar no desenvolvimento do projeto: podemos ser específicos demais e não conseguirmos reaproveitar código ou podemos ser específicos de menos e acabar aplicando estilos em lugares que não gostaríamos.

UTILIZANDO DIVs, IDs e CLASS:

* **DIV:**

O elemento <div> é um container genérico para conteúdo de fluxo, que de certa forma não representa nada. Ele pode ser utilizado para agrupar elementos para fins de estilos (usando class ou id). Ou seja, A tag div não é mais do que um container para outras tags. A intensão desta tag é criar subcontainers para a sua página HTML.

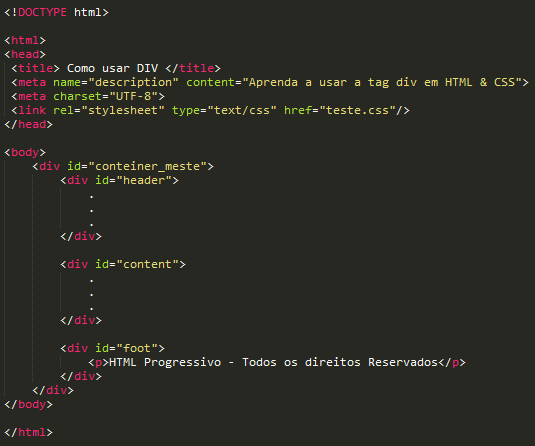
* **ID:**

As IDs são uma forma de identificar um elemento, e devem ser ÚNICAS para cada elemento. É como se fossem impressões digitais de nossos dedos ou RGs. Através delas, pode-se atribuir formatação a um elemento em especial.

* **CLASS:**

As classes são uma forma de identificar um grupo de elementos. Através delas, pode-se atribuir formatação a VÁRIOS elementos de uma vez.

Vamos ver a utilização destes recursos e tentar entende-los.



Repare que criamos quatro “DIVs”, a primeira sendo o contêiner mestre, a segunda para o cabeçalho, na sequencia uma div referente ao conteúdo e por fim uma div para o rodapé. Com isso podemos personalizar e ajustar estas div cada uma de uma forma diferente conforme o que desejar. Veja no CSS como podemos referenciar um objeto com ID.

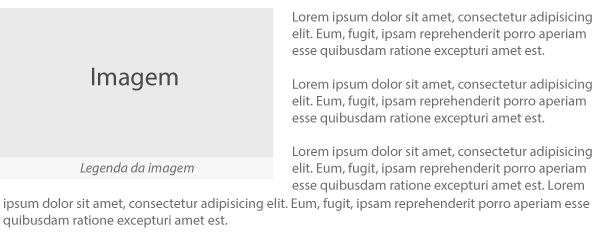
O parâmetro id pode ser usado em todos os objetos e para referencia-los no CSS basta utilizar um símbolo de # e o nome do objeto dado. Da mesma forma você deve fazer para utilizar o “class”, no entanto, ao invés de utilizar o símbolo de #, você deve troca-lo por um **ponto (.)**

Aplique os códigos HTML e CSS mostrados para que possa entender e ver a aplicação das divs, ids e class.



# Propriedade Float e Clear

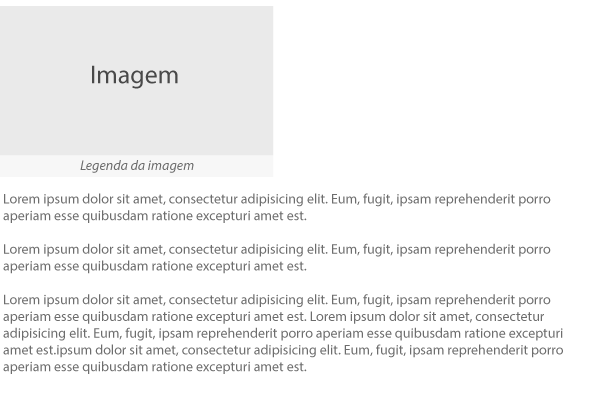
Algo comum ao lermos alguma matéria em revistas ou jornais é uma imagem relacionada ao texto estar posicionada de um lado e, o conteúdo textual contornar a imagem. Algo parecido com isso:



E se precisássemos desenvolver um *template* parecido com a imagem acima. Provavelmente começaríamos com um código HTML assim:

1. <div>
2. <figure>
3. <img src="img/imagem.jpg" alt="Imagem Xis">
4. <figcaption>Legenda da imagem"</figcaption>
5. </figure>
6. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum, fugit, ipsam reprehenderit porro aperiam esse quibusdam ratione excepturi amet est.</p>
7. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum, fugit, ipsam reprehenderit porro aperiam esse quibusdam ratione excepturi amet est.</p>
8. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum, fugit, ipsam reprehenderit porro aperiam esse quibusdam ratione excepturi amet est. Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum, fugit, ipsam reprehenderit porro aperiam esse quibusdam ratione excepturi amet est</p>
9. </div>

O que teríamos seria algo assim:



Por que isso? O que acontece é que a tag **figure** ocupa toda a largura da página e entra no empilhamento do documento/página, o que não permite que outros elementos apareçam ao seu lado.

É aí que entra o *float*. Em resumo, essa propriedade permite que tiremos um elemento do fluxo vertical da página e automaticamente faz com que o conteúdo que venha a seguir *flutue* ao seu redor.

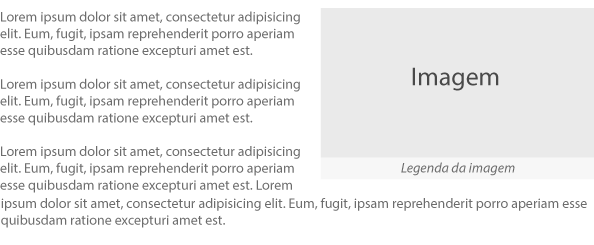
O *CSS* ficaria assim:

1. figure {
2. float: left;
3. margin: 0 10px 10px 0;
4. }

O que fizemos foi *flutuar* a tag **figure** para o lado esquerdo e, colocar uma margem inferior e direita para que os parágrafos não fiquem grudados na imagem. Com isso conseguiríamos reproduzir um layout igual ao da primeira imagem. Poderíamos até variar e colocar nossa imagem do lado direito:

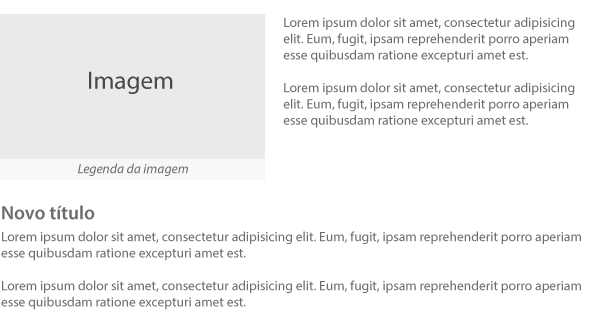
1. figure {
2. float: right;
3. margin: 0 0 10px 10px;
4. }

Aqui mudamos a direção que a tag **figure** vai flutuar e alteramos a margem do lado direito para o lado esquerdo. E teríamos um resultado assim:



### **A propriedade *clear***

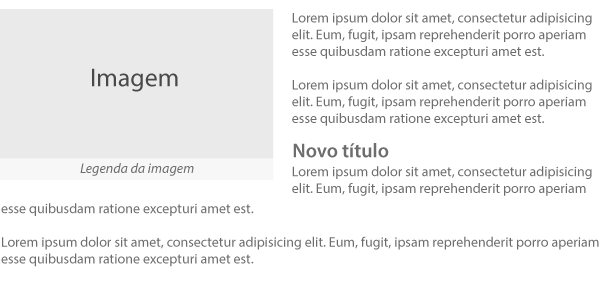
Continuando com o nosso exemplo anterior, imagine que agora precisássemos inserir um novo título com mais parágrafos abaixo da imagem. Algo parecido com isso: (para fins de exemplo, diminui o conteúdo textual ao redor da imagem)



Para representar esse conteúdo, teríamos um *HTML* semelhante a esse:

1. <div>
2. <figure>
3. <img src="img/imagem.jpg" alt="Imagem Xis">
4. <figcaption>Legenda da imagem"</figcaption>
5. </figure>
6. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum, fugit, ipsam reprehenderit porro aperiam esse quibusdam ratione excepturi amet est.</p>
7. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum, fugit, ipsam reprehenderit porro aperiam esse quibusdam ratione excepturi amet est.</p>
8. <h3>Novo título</h3>
9. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum, fugit, ipsam reprehenderit porro aperiam esse quibusdam ratione excepturi amet est.</p>
10. <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Eum, fugit, ipsam reprehenderit porro aperiam esse quibusdam ratione excepturi amet est.</p>
11. </div>

No entanto, chegaríamos num resultado assim:



Por que isso? Como definimos que a tag **figure** flutuaria à esquerda, saindo assim do fluxo vertical da página, todo conteúdo que vem após ela começa a preencher o espaço ao redor da imagem. O que acontece é que os parágrafos que vem logo após a tag **figure** são menores que a altura da imagem, fazendo com que o título (tag **h3**) ao invés de ser renderizada abaixo da imagem, apareça ao lado dela e seguindo o fluxo do documento.

É aí que entra a propriedade *clear*. Ela tem a função de controlar o comportamento de elementos que apareçam no fluxo do documento após determinado elemento que possue a propriedade *float*. Em outras palavras, ela especifica se um elemento deve ser posicionado ao lado de elementos com *float* ou se devem ser colocados abaixo deles. A propriedade aceita 4 valores:

* **left**: Elemento é empurrado para baixo de elementos com *float left*;
* **right**: Elemento é empurrado para baixo de elementos com *float right*;
* **both**: Elemento é empurrado para baixo de elementos com *float left ou right*;
* **none**: Elemento não é empurrado para baixo de elementos com *float*.

O *CSS* ficaria assim:

1. figure {
2. float: left;
3. margin: 0 10px 10px 0;
4. }
6. h3 { clear: both; }
7. /\* Pelo fato da tag figure utilizar float left, aqui também poderíamos usar clear: left; \*/

E com isso, conseguiríamos fazer com que o título (tag **h3**) fosse empurrado para baixo, sendo renderizado abaixo da tag **figure** e ficando igual ao layout que queríamos.

CRIANDO FORMULÁRIOS:

**Form:** O elemento <form> ao qual o elemento <input> está associado (seu formulário dono). O valor do atributo deve ser um id de um elemento [<form>](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/form). Este atributo permite que você coloque elementos <input> em qualquer lugar num documento, não apenas como descendentes de seus elementos <form>. Um <input> só pode estar associado a um único formulário.

**Input:** O elemento HTML<input> é usado para criar controles interativos para formulários baseados na web para receber dados do usuário. A semântica de um <input> varia consideravelmente dependendo do valor de seu atributo type.

O tipo de controle a ser exibido. O tipo padrão é text, se este atributo não for especificado. Os valores possíveis são:

**button**: Um botão sem comportamento padrão.

**checkbox**: Uma caixa de marcação. Você deve usar o atributo value para definir o valor enviado por este item. Use o atributo checked para indicar se o item está selecionado por padrão. Você também pode usar o atributo indeterminate para indicar que a caixa de marcação está em um estado indeterminado (na maioria das plataformas, isso desenha uma linha horizontal cortando a caixa).

**color**: Um controle para especificar cores. A interface de um seletor de cores não tem nenhuma funcionalidade obrigatória a não ser aceitar cores simples em texto ([mais informações — em inglês](http://www.w3.org/TR/html5/forms.html#color-state-(type=color))).

**date**: Um controle para inserir uma data (ano, mês e dia, sem horário).

**datetime**: Um controle para inserir data e horário (hora, minuto, segundo e fração de segundo) baseado no fuso horário UTC.

**datetime**-**local**: Um controle para inserir data e horário, sem fuso horário.

**email**: Um campo para editar um endereço de e-mail. O valor do campo é validado para estar vazio ou ter um único endereço de e-mail válido antes de ser enviado. As pseudoclasses CSS [:valid](https://developer.mozilla.org/pt-BR/docs/Web/CSS/:valid) e [:invalid](https://developer.mozilla.org/pt-BR/docs/Web/CSS/:invalid) são aplicadas apropriadamente.

**file**: Um controle que permite ao usuário selecionar um arquivo. Use o atributoaccept para definir os tipos de arquivo que o controle pode selecionar.

**hidden**: Um controle que não é exibido mas cujo valor é enviado ao servidor.

**image**: Um botão gráfico para enviar o formulário. Você deve usar o atributo src para definir a fonte da imagem e o atributo alt para definir um texto alternativo. Você pode usar os atributos height e width para definir o tamanho da imagem em pixels.

**month**: Um controle para inserir mês e ano, sem fuso horário.

**number**: Um controle para inserir um número de ponto flutuante.

**password**: Um campo de texto com uma só linha cujo valor é obscurecido. Use o atributo maxlength para especificar o comprimento máximo do valor que pode ser inserido.

**radio**: Um botão de escolha. Você deve usar o atributo value para definir o valor a ser enviado por este item. Use o atributo checked para indicar se este item deve estar selecionado por padrão. Botões de escolha que têm o mesmo valor para o atributo name estão no mesmo "grupo de botões de escolha"; apenas um botão de escolha no grupo pode estar selecionado de cada vez..

**range**: Um controle para inserir um número cujo valor exato não é importante. Este tipo de controle usa os seguintes valores padrão se os atributos correspondentes não forem especificados:

min: 0

max: 100

value: min + (max-min)/2, ou min se max for menos que min

step: 1

**reset**: Um botão que faz o conteúdo do formulário voltar a ter seus valores padrão.

**search**: Um campo de texto com uma só linha para digitar termos de busca; quebras de linha são automaticamente removidas do valor entrado.

**submit**: Um botão que envia o formulário.

**tel**: Um controle para inserir um número de telefone; quebras de linha são automaticamente removidas do valor entrado, mas nenhuma outra sintaxe é imposta. Você pode usar atributos como pattern e maxlength para restringir os valores inseridos no controle. As pseudoclasses CSS [:valid](https://developer.mozilla.org/pt-BR/docs/Web/CSS/:valid) e [:invalid](https://developer.mozilla.org/pt-BR/docs/Web/CSS/:invalid) são aplicadas apropriadamente.

**text**: Um campo de texto com uma só linha; quebras de linha são automaticamente removidas do valor entrado.

**time**: Um controle para inserir um horário sem fuso horário.

**url**: Um campo para editar uma URL. O valor inserido é validado para ser vazio ou uma URL absoluta válida antes de ser enviado. Quebras de linha e espaços em branco antes e após o valor inserido são automaticamente removidos. Você pode usar atributos como pattern e maxlength para restringir os valores inseridos no controle. As pseudoclasses CSS [:valid](https://developer.mozilla.org/pt-BR/docs/Web/CSS/:valid) e [:invalid](https://developer.mozilla.org/pt-BR/docs/Web/CSS/:invalid) são aplicadas apropriadamente.

**week**: Um controle para inserir uma data consistindo de ano da semana e número da semana sem fuso horário.

OUTROS ATRIBUTOS PARA USARMOS EM FORMULARIOS:

**max**

O valor máximo (numérico ou data) para este item, que não deve ser menor que seu valor mínimo (atributo **min**).

**maxlength**

Se o valor do atributo **type** for text, email, search, password, tel ou url, este atributo especifica o número máximo de caracteres (em pontos de código Unicode) que o usuário pode inserir; para outros tipos de controle, este atributo é ignorado. Seu valor pode exceder o do atributo **size**. Se não for especificado, o usuário pode inserir uma quantidade ilimitada de caracteres. Especificar um valor negativo resulta no comportamento padrão, isto é, o usuário vai poder inserir um número ilimitado de caracteres. Esta regra só é validada quando o valor do atributo é modificado.

**min**

O valor mínimo (númerico ou data) para este item, que não deve ser maior do que seu máximo (atributo **max**).

**minlength**

Se o valor do atributo **type** for text, email, search, password, tel ou url, este atributo especifica o número mínimo de caracteres (em pontos de código Unicode) que o usuário pode inserir; para outros tipos de controle, este atributo é ignorado.

**multiple**

Este atributo booleano indica se o usuário pode inserir mais de um valor. Ele é aplicado quando o atributo **type** está definido como email ou file; em outros casos, é ignorado.

**name**

O nome do controle, que é enviado junto com os dados do formulário.

**pattern**

Uma expressão regular usada para validar o valor do controle. O padrão deve casar com o valor completo da entrada, não apenas uma parte. Use o atributo **title** para descrever o padrão para ajudar o usuário. Este atributo é aplicado quando o valor do atributo **type** é text, search, tel, url ou email; caso contrário ele é ignorado. A linguagem da expressão regular é a mesma do JavaScript. O padrão não deve estar entre barras.

**placeholder**

Uma dica para o usuário do que ele pode inserir no controle. O texto do atributo não deve conter quebras de linha. Este atributo é aplicado quando o valor do atributo **type** é text, search, tel, url ou email; caso contrário, ele é ignorado.

**readonly**

Este atributo booleano indica que o usuário não pode modificar o valor do controle.

 Este atributo é ignorado se o valor do atributo **type** for hidden, range, color, checkbox, radio, file ou um tipo de botão.

**required**

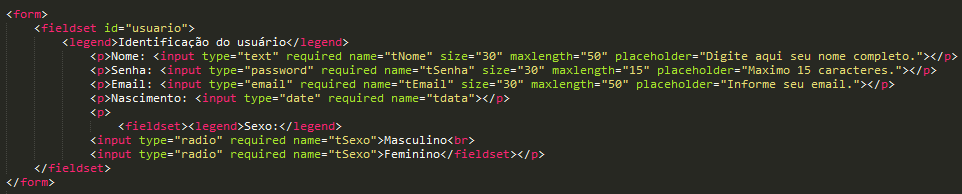
Este atributo especifica que o usuário deve preencher o campo com um valor antes de enviar o formulário. Ele não pode ser usado quando o atributo **type** está definido como hidden, image ou um tipo de botão (submit, reset ou button). As pseudoclasses CSS [:optional](https://developer.mozilla.org/pt-BR/docs/Web/CSS/:optional) e [:required](https://developer.mozilla.org/pt-BR/docs/Web/CSS/:required) serão aplicadas ao campo como for apropriado.

**selectionDirection**

A direção na qual a seleção ocorreu. Vale forward se a seleção foi feita no mesmo sentido da escrita do idioma usado ou backward se a seleção foi feita na direção oposta. Pode ainda ser none se a direção da seleção não for conhecida.

**size**

O tamanho inicial do controle. Este valor é em pixels a não ser que o valor do atributo **type**seja text ou password. Neste caso, este valor é um número inteiro de caracteres. A partir do HTML5, este atributo é aplicado apenas quando o atributo **type** está definido como text, search, tel, url, email ou password; caso contrário, ele é ignorado. Além disso, o tamanho deve ser maior que zero. Se você não especificar um tamanho, o valor padrão 20 é usado.

Dentro de um formulário podem existir inúmeros objetos de tipos diferentes. Vamos criar alguns deles com os exemplos abaixo.

No exemplo acima, a tag <form> é responsável por determinar tudo o que é pertencente ao formulário, todos os objetos que sejam pertinentes ao formulário devem ser contidos dentro da abertura e fechamento da tag <form>.

A seguir da <form> temos a <fieldset> onde a função desta tag é criar uma borda em volta dos objetos contidos dentro dela. Em seguida, temos a tag <legend> com a função de criar uma legenda para o formulário, como se fosse um título a que pertence ele. Esta legenda ficará dentro da fieldset.

A seguir inserimos de fato os objetos de um formulário, através da tag <input> adicionamos os textbox, cada objeto do tipo “input” trará consigo dois parâmetros: *type* e *name.* No “type” você pode determinar com qual tipo de informação estará trabalhando. Conforme o exemplo acima, repare que em cada linha utilizamos um type diferente. No caso do “text” definimos que será uma caixa de texto padrão que aceita letras, números, caracteres especiais e etc. No type “password” a caixa de texto passa a não exibir os caracteres digitados, ela passa a realmente ocultar o que é digitado como se fosse um campo de senha. O type “email” visualmente vai parecer com o “text” no entanto, quando for clicado para enviar o formulário, ele fará uma verificação se o campo especifico está dentro dos padrões de um email. Temos também o type “date” onde o campo ficará com um tipo de calendário para inserção de datas. Temos também o type “radio”, onde este tipo cria um objeto de radio button para escolhas.

Os outros parâmetros dentro do código servem cada um para uma função. Veja:

* O parâmetro “size” serve para definir o tamanho do campo de texto.
* O parâmetro “required” serve para definir quais os campos são obrigatórios e quais não, neste caso, todos os campos que estiverem com o required tornam-se obrigatórios o preenchimento.
* O parâmetro “placeholder” serve para que você possa adicionar um breve texto informativo dentro da caixa de texto. Como se fosse uma dica ao usuário. Ao selecionar e começar a digitar no campo especifico o texto de placeholder desaparece. **Nota:** Não use o atributo placeholder no lugar de um elemento [<label>](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/label). Os propósitos de cada um são diferentes: o atributo [<label>](https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/label) descreve o papel o elemento do formulário, isto é, ele indica que tipo de informação é esperada. Já o atributo placeholder é uma dica sobre o formato que o conteúdo deveria ter. Há casos em que o atributo placeholder nunca é exibido para o usuário, portanto o formulário deve ser inteligível sem ele.
* O parâmetro “maxlenght” serve para você limitar a quantidade de caracteres que podem ser digitados dentro daquele campo. Por exemplo, se você quer definir que uma senha deve ter no máximo 10 letras, pode definir isto através do parâmetro maxlenght.

O primeiro passo para criarmos um formulário em HTML é usar a *tag* "*form*":

<div class="container">

<form>

...

</form>

</div>

Vamos aos poucos implementando-a. Começando pelos espaços onde o usuário colocará seu nome, seu e-mail:

<form>

<label>Nome:</label>

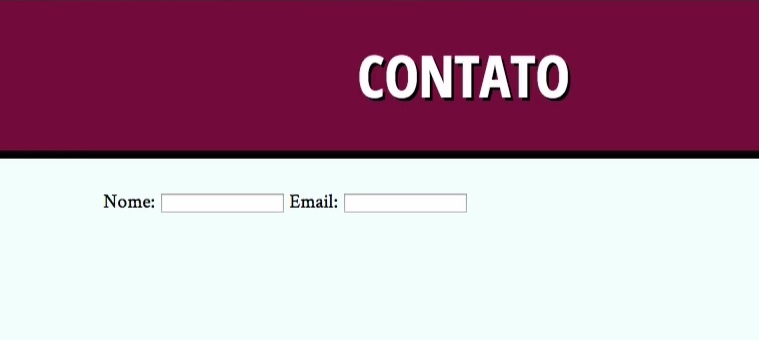
<input>

<label>Email:</label>

<input>

</form>

Até agora nosso formulário está assim:



O usuário também poderá escolher o assunto do qual se trata a mensagem. Isso será feito por meio de botões:

<form>

<label>Nome:</label>

<input>

<label>Email:</label>

<input>

<input type="radio">

<label>Consultoria</label>

</form>

Antes de continuarmos, note que o navegador não saberá a que campo se referem os *inputs*. Para isso a *tag* "*label*" deve possuir o atributo"*for*" e o *input* o atributo "*id*", exatamente para identificar a qual campo ele se refere.

<form>

<label for="nome">Nome:</label>

<input id="nome">

<label for="email">Email:</label>

<input id="email">

<input type="radio" id="consultoria">

<label for="consultoria">Consultoria</label>

</form>

Visualmente nada muda, porém muda-se a semântica e a acessibilidade, o que ajuda o usuário a selecionar os campos.

Se quisermos, também podemos deixar o *input* dentro do *label como no exemplo abaixo*:

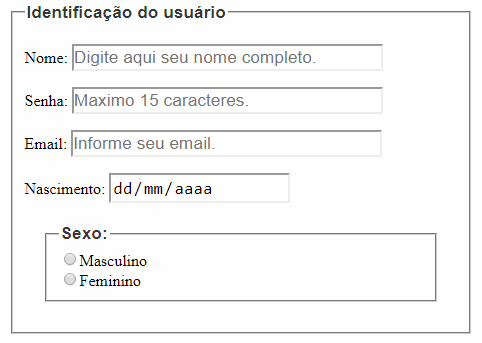
<label>

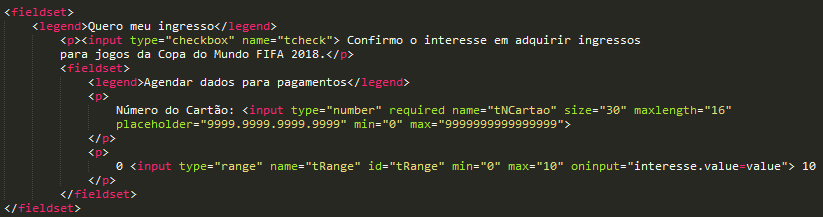
<input type="radio">

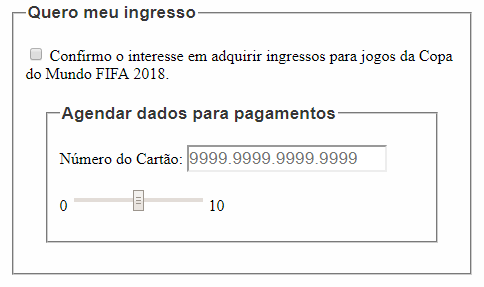
Consultoria

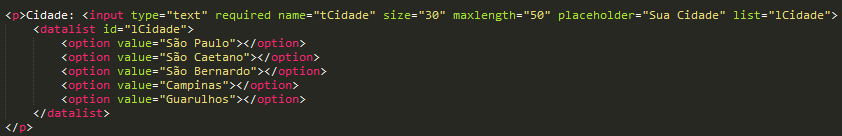
</label>

Aplique o código acima e veja o que é criado a partir dele, veja qual o resultado obtido:



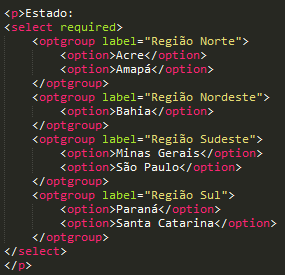
Além dos tipos de objetos vistos no exemplo acima ainda temos os types: “number” onde dentro do campo de texto serão aceitos apenas caracteres numéricos. Temos o type “checkbox” onde será criado um checkbox podendo selecionar vários deles. Por fim, temos os objetos input do type “range”, onde este objeto cria uma barra de rolagem vertical que tem valores atribuídos a ele. Veja abaixo o código para cada um destes objetos e como fica sua aplicação.



Dentro de um objeto do tipo “text” podemos adicionar um <datalist>. Este datalist tem o objetivo de deixar pre-definido alguns dados para facilitar na hora que os usuários estiverem digitando, ele já traz algumas opções. Para isso, utilizamos o datalista da seguinte forma:

Repare que na primeira linha do código criamos um campo de texto comum, como vimos exemplos antes, no entanto, no ultimo parâmetro desta linha utilizamos um parâmetro chamado “list” e dentro deste list colocamos o valor de “lcidades”, logo abaixo criamos uma lista, adicionamos alguns valores a esta lista e através do id demos o nome para esta lista de “lcidades”. Ou seja, na primeira linha quando criamos o parâmetro ( list=”lcidades” ) especificamos de qual local ele deveria buscar as opções cidades. Este datalist serve como se fosse uma reserva de informações de cidades e quando o usuário digitar as cidades no campo de texto, ele começa a fazer o filtro de acordo com os dados que estão dentro do datalist. Não se esqueça, o nome que você der para o datalist através do id deve ser o mesmo mencionado no parâmetro list do input que deseja utilizar.

Em um formulário também temos a tag <select>, esta tag criar um objeto que traz opções que foram pré-definidas. Este recurso por exemplo é muito utilizado em estados de um país.

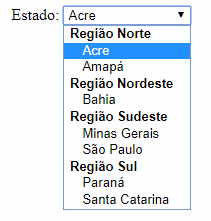
 Vamos as especificações de cada uma das tags utilizadas no código ao lado.

A tag <select> serve para criar o objeto de opções e delimitar que tudo o que estiver dentro dela é pertencente a este objeto.

A tag <optgroup> serve para criarmos um título de opções, esta tag deve estar acompanhado do parâmetro “label” e dentro deste “label” adicionar qual o título queremos. Ele serve apenas como uma divisão, não pode ser selecionável, no exemplo acima é apenas para, por exemplo, fazer a divisão dos estados de acordo com a região geográfica de cada um deles.

Já a tag <option> esta é a responsável por adicionar as opções que de fato podem ser escolhidas, neste caso, os estados em si, para que o usuário possa escolher o seu determinado estado.

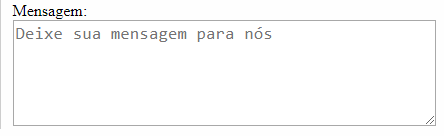
Veja abaixo o resultado obtido:



Podemos também utilizar o objeto <textarea> onde será criado uma caixa de texto que permite mais caracteres, através dos parâmetros “cols” e “rows” você pode determinar o tamanho dela.

Mensagem: <textarea name=”tmsg” cols=“40” rows=”5” placeholder=”Deixe sua mensagem para nós”>

E com o código acima criamos a seguinte objeto:



Após a construção do formulário, devemos adicionar os botões de confirmação e/ou cancelamento. Veja como fazemos para adicionar estes botões.



No código acima estamos criando dois botões. No primeiro colocamos o type como “submit” pois este tipo tem sua função pré-definida que enviar os dados do formulário, o parâmetro “value” é definido o texto que vai aparecer dentro do botão.

No segundo botão também criamos um com um tipo já pré-definido, no caso o tipo “reset” faz a limpeza dos textos no formulário.

Caso queira criar um botão inicialmente sem nenhum tipo de comportamento atribuído a ele basta no type colocar como “button”. Desta forma será criado um botão sem nenhum comportamento vinculado a ele.

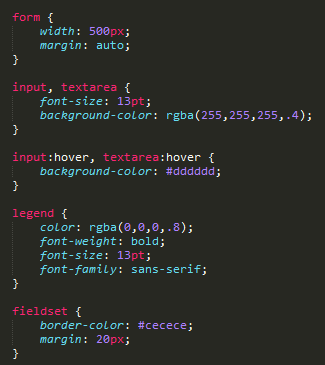
Podemos personalizar nossos formulários adicionando cores de fundo aos objetos, cores diferenciadas as letras, efeitos de transação ao passar o mouse por cima de uma caixa de texto, o tamanho do formulário e afins. Vale lembrar que todas essas personalizações são feitas através do **CSS** e para que elas surtam efeito sua página CSS deve estar devidamente configurada e vinculada à página HTML. Caso tenha dúvidas ou tenha esquecido de como fazer isso, dê uma olhada no material novamente. Abaixo uma amostra do código CSS de um exemplo que pode ser utilizado para modificar as configurações de estética da página.

Lembre-se, no CSS você deve referenciar qual tipo de objeto que personalizar e fazer as devidas configurações, mudanças e personalizações para este objeto.

No código abaixo você verá que criamos um objeto chamado form e que este objeto terá um tamanho de 500px. Para os objetos input e textarea definimos o tamanho das fontes para cada um deles. Abaixo também poderá ver um objeto com um parâmetro chamado “hover”, este parâmetro chamado **hover** permite que ao passar o mouse por cima de determinado objeto ele tenha um comportamento. Neste caso foi definido que ao passar o mouse sobre os objetos do tipo input e textarea os mesmos terão uma cor de background diferente de quando o mouse não estiver sobre os mesmos. Lembre-se é por conta do **hover** que isto é possível e o **hover** não se limita a apenas objetos do tipo input ou textarea. Você pode utiliza-lo para links, imagens, menus e inúmeros outros objetos.

No código abaixo também configuramos a estilização dos objetos legend e fieldset. Cada um com suas particularidades de acordo com o desejo do desenvolvedor ou de seu cliente.

*Faça os testes e verifique as inúmeras possibilidades que pode utilizar.*



Lembre-se de que para todo objeto que você inserir no seu HTML é aconselhável nomeá-los, isso facilitará posteriormente quando necessitar utiliza-los ou referencia-los em outros momentos e locais.

Afinal, o que ganhamos quando usamos um tipo mais específico de campo de formulário? Quando usamos um tipo mais específico de campo, estamos agregando **semântica** ao nosso formulário, e isso traz diversas vantagens na prática.

Uma primeira vantagem é a **validação** dos campos. O campo do tipo email garante que o usuário deve digitar um e-mail válido na hora de enviar o formulário. O campo do tipo number garante que o valor digitado será numérico.

Outra grande vantagem é uma **usabilidade** melhor do formulário, principalmente em dispositivos móveis. Quando usamos um tipo date, por exemplo, o navegador pode mostrar um controle para selecionarmos uma data. Ou, quando usamos um tipo number, o dispositivo móvel pode mostrar um teclado numérico, facilitando a digitação.

Além dessas vantagens, temos todas as vantagens de ter uma página mais semântica: mais acessibilidade, melhor indexação pelas ferramentas de busca etc.

**O atributo pattern**

Vimos que o HTML5 trouxe validação para formulários sem a necessidade de JavaScript, com novos tipos e o atributo required. Além disso, temos também o atributo pattern.

No atributo pattern, podemos especificar qual padrão deve seguir um determinado campo, para que o usuário preencha-o corretamente. Especificamos esse padrão usando uma linguagem chamada expressão regular. Para garantir que um campo só tenha letras e números, por exemplo, podemos fazer:

<input pattern="[A-Za-z0-9]\*">

Repare que colocamos os conjuntos de caracteres permitidos entre colchetes (letras maiúsculas, minúsculas e dígitos, nessa ordem) e um asterisco depois. Esse asterisco significa que podemos repetir quantas vezes quisermos um caractere desse conjunto. Se quisermos garantir uma quantidade mínima de letras e números, podemos fazer

<input pattern="[A-Za-z0-9]{8,}">

Com esse padrão, agora nosso usuário precisa digitar pelo menos 8 letras ou números para a entrada ser válida.

Com base nisso, altere o campo "Nome" para que o usuário só possa entrar um nome contendo letras (maiúsculas e minúsculas), espaços e apóstrofo, com no mínimo 4 símbolos. Como resposta, forneça o código do <input> alterado.

**Descrevendo o pattern**

Fizemos um padrão personalizado para o campo "Nome", garantindo dados mais confiáveis dos nossos visitantes. Porém, quando um dado inválido é inserido nesse campo, a mensagem de erro é muito genérica! Vamos melhorar isso.

Num campo como o "Nome", com o atributo pattern, podemos usar o atributo titlepara descrever esse padrão. Por exemplo:

<input pattern="[0-9]\*" title="Preencha somente com números">