HTML5 && CSS3 – Alura

Parte 1 – A primeira página da Web

1. Aula 1 – Marcação do Primeiro Texto:

1.1. Uma página web tem seu conteúdo em HTML: “Hyper text marcation language” – linguagem de marcação de texto.

1.2. O formato de uma tag é *<tag>*

1.3. Para utilizar uma tag, é necessário indicarmos onde se iniciam e onde terminam. Para indicar a finalização da tag, usamos ela novamente, mas somada a uma “/”, ou seja, *<h1>Título</h1>*.

1.4. As tags que usaremos nesse texto base da primeira aula são:

1.3.1 <h> (heading): Título do conteúdo – Possui 6 níveis:

1.3.1.1 <h1>: Para o primeiro título da página;

1.3.1.2 <h2>:

1.3.1.3 <h3>:

1.3.1.4 <h4>:

1.3.1.5 <h5>:

1.3.1.6 <h6>:

1.3.2 <p> (paragraph): Usado para colocar parágrafos de textos.

1.3.3 (forte/negrito): É uma tag semântica usada para deixar as palavras ou frases dentro delas com uma tonalidade mais forte, dando destaque.

1.3.4 (ênfase): Usada para dar ênfase em um parágrafo u palavra, ao utilizar essa tag, você determina que tudo o que estiver dentro está em itálico.

2. Aula 2 – Separando o Conteúdo e Informações:

1.1. Para que o navegador renderize a nossa página da forma correta, não basta que coloquemos .html na hora de salvar o arquivo, existem outras tags de marcação que precisamos acrescentar no nosso código para que isso se torne explícito.

1.2. Nós precisamos deixar claro para o navegador que estamos trabalhando cm HTML5, portanto, temos tags específicas para ter certeza que o navegador não irá interpretar a nossa página como HTML3.

1.3. As tags que utilizaremos nessa aula são:

1.3.1. <!DOCTYPE html>: Esta tag declara para o navegador que estamos utilizando a última versão do html, dessa forma ele irá interpretar a página corretamente. Não precisa ser fechada como as outras tags por não ser uma tag de conteúdo.

1.3.2. <html>: Tag estrutural que serve para marcar tudo o que é o html que vai ser renderizado no navegador a partir dessa página. Como as outras tags de conteúdo, precisamos fechar essa tag, colocando </html> no final do documento, literalmente depois de todo o resto que está no meio da página.

1.3.2.1. Dentro desta tag colocamos um parâmetro para identificar em qual idioma essa página se encontra. O parâmetro que utilizamos é o *lang="pt-br"*. Parâmetros são colocados dentro dos <> junto com as tags correspondentes.

1.3.3. <meta>: Tag usada para indicar qual o idioma e acentuações estamos utilizando nessa página, dessa forma, o navegador pode exibi-la corretamente independentemente do navegador e de onde a página for aberta.

1.3.3.1. O parâmetro que vamos utilazar para que as acentuações fiquem corretas na página é o *charset="UTF -8"*- este é o dicionário que estamos utilizando para a nossa página, dicionário utf-8 especificado pelo parâmetro charset.

1.3.3.2. Essa tag no geral, independentemente do parâmetro que esteja junto com ela, será sempre colocada abaixo da tag <html>.

1.3.3.3. Essa não é uma tag de conteúdo, portando, não precisa ser fechada em algum momento.

1.3.4. <title>: Tag usada para determinar o título de uma página, o que fica na guia, lá no topo do navegador.

1.3.5. <head>: Tag usada para classificar e colocar todas as tags informativas para a página, como a tag <meta>, <title>, dentre outras.

1.3.6. <body>: Tag usada para colocar todo o conteúdo que será exibido na página, como o <h1>, <p>, dentre outras tags e conteúdos escritos.

3. <u>Aula 3 – Trabalhando com CSS</u>:

1.1. Taduzindo: Folha de estilo em cascata.

1.2. Para adicionar uma estilização inline, ou seja, aquela que se coloca dentro da tag específica e que será aplicada somente a ela, é necessário colocar o parâmetro *style="";* dentro da tag desejada, como um parágrafo (<p>), por exemplo.

1.3. Dentro desse parâmetro, podemos colocar a estilização desejada, como alterar o tamanho da fonte com *font-size=: ;*, por exemplo. O tamanho padrão da fonte que o navegar aplica automaticamente é 16px (pixels) que equivale a 1em (tamanho medido a partir da altura da letra "m" minúscula de uma fonte específica) ou 1ex (igual ao "em", mas se refere à letra "x") ou 12pt (points – usado no word), portanto, essas alturas/tamanhos podem ser alterados utilizando esse parâmetro e comando de CSS3 mostrados.

1.4. Utilizando o parâmetro *text-align: ;* podemos ajustar o alinhamento de um texto. No caso de um título, podemos colocar dentro da tag <h1> o style="text-align: center;", ficando assim: <h1 style="text-align: center;">. Utilizando esse alinhamento, o título passa a ficar no centro da página, independente do tamanho da janela do navegador.

1.5. Existem 3 formas de configurar o CSS:

1.5.1. Inline: Já falado acima, onde a estilização é colocada especificamente em cada tag desejada. A técnica menos utilizada e que demanda mais trabalho para alteração de vários documentos. Definitivamente não é recomendada a sua utilização. Caso não possua mais de 1 página, ou mesmo se tiver apenas uma, utilize sempre os métodos abaixo. Deixa o código mais limpo e melhor para ler, além de facilitar qualquer tipo de alteração futura que queira fazer na sua estilização, sem grandes dores de cabeça por ter que alterar tag por tag até a finalização.

1.5.2. Local: Uma tag <style></style> que é colcoada dentro do <head> de um <html> e lá iremos configurar cada tag do html em conjunto, ou seja, se colocarmos <style>p{text-align: center; color: blue; font-size: 20px}</style>, a alteração especificada irá acontecer em todos os parágrafos que tenham a tag <p>. Método bastante utilizado e até que recomendado caso não possua muitas páginas em um site.

1.5.3. Externo: Toda a estilização é feita em um documento separado cuja a extensão é .css e é linkado ao documento .html pela tag <link>. Dessa forma, podemos fazer toda a estilização de um site com apenas 1 documento. Como geralmente utilizamos o mesmo padrão de estilização

para todo o conjunto de páginas de um site, esse é o método mais utilizado e mais recomendado, uma vez que, caso haja a necessidade de se fazer uma alteração na estilização, não será necessário abrir o documento de todas as páginas e alterar um por um até terminar, basta alterar aquele documento .css externo que as alterações se aplicarão a todas as suas páginas web imediatamente após o salvamento.

1.5.3.1. Dentro da tag <link> precisamos colocar alguns parâmetros para fazer o link com o doc .css:

1. rel=”stylesheet” (relação)
2. href=”style.css” (endereço de referência – esse endereço é o padrão, caso não esteja na mesma pasta que a página é necessário alterar. Para alterar, basta apagar tudo dentre os “” e apertar ctrl+space. Ele vai abrir uma janela onde você pode navegar e escolher a folha de estilo que deseja usar para aquele site).

1.5.4. Utilizando o comando *bakcground-color: ;* nós podemos alterar a cor do fundo de qualquer coisa, desde um parágrafo/frase/título até o fundo ba página toda.

1.5.5. Podemos alterar qualquer estilo de uma tag específica utilizando prioridades de busca em CSS. Por exemplo, se vc quiser alterar a cor de uma frase específica que está dentro de uma tag que está dentro de uma tag , assim: *Nossa missão é: ”Proporcionar auto-estima e qualidade de vida aos clientes”*. Nós podemos fazer a seguinte classificação no CSS: *em strong{color: red}*. Desse modo, ele irá buscar e alterar a cor de todos os strong’s que estejam dentro dos em’s, se tiver algum fora, continuará do mesmo jeito que antes.

4. <u>Aula 4 - Esilizando Imagens</u>:

1.1. Nós podemos adicionar identificadores (id=”nome”) para qualquer tag especificamente, para selecionar qualquer uma que queira e alterar seu estilo num arquivo .css separado sem modificar as outras, vamos supor que vc tenha um parágrafo e coloque um id=”missao”, quando for estilizar ele no css, basta referenciar como #missão{font-size: 20px;}, e ele será alterado para o que quiser.

1.2. Para adicionar uma imagem nós utilizamos a tag <img>.

1.3. Como a imagem é um arquivo externo, nós precisamos dizer a onde ela está, dessa forma, adicionando o parâmetro src=”caminho”

1.4. Além do caminho, também colocamos o parâmetro alt=”descrição”, para dizer o que é essa foto, se é a foto de um banner, perfil, dentre outras coisas. Isso é importante para quando der algum problema e não for possível carregar a imagem, pois é o texto que foi colocado lá que irá aparecer no lugar.

1.5.Para fazer alterações em css de uma imagem, utilizamos uma id=”nome”, colocamos #nome{} e então podemos alterar como quisermos. Obs.: O símbolo “#” é chamado de “tralha”, não jogo da velha e nem hashtag.

1.6. Podemos colocar largura de uma altura em 100% para ela ocupar sempre a largura toda da tela, independentemente do tamanho da janela do navegador. Usamos *width: 100%;* pra isso.

1.7. Podemos criar uma borda em qualquer elemento usando a marcação *border: tamanho tipo cor*, exemplo: border: 10px solid black. A borda é a finalização do elemento.

1.8. O espaçamento interno, que fica entre a bora e o conteúdo do elemento (seja uma foto, ou texto) é chamado de *padding* e podemos criar vários tipos de espaçamentos, somente em cima, somente em baixo, na direita ou esquerda, além é claro, de em todos os lados ao mesmo tempo. Por exemplo: padding: 20px;. Com essa marcação, o nosso elemento teria um respiro de 20 pixels entre a borda e o texto digitado em todas as direções. Para criar em direções específicas basta colocar *-local*. Exemplo: padding-top: 20px; - cria um espaçamento entre o conteúdo e a borda de 20 pixels na parte superior da box. O mesmo ocorre se colocar -bottom, -left e -right.

1.9. O espaçamento inter, que fica entre a box e o resto do conteúdo da página, é chamada de *margin*, literalmente uma margem entre a box e o conteúdo. O esquema de direcionamento e tamanho funciona igual ao *padding*, usando top, left, 20px, 60px, etc...

5. <u>Aula 5 – Listas e divisões de Conteúdo</u>:

1.1. Existem duas tags para criação de listas:

1.1.1. (ordered list): Listas ordenadas, ou seja, numeradas, ou alfabéticas ou por números romanos todos em letras maiúsculas ou não.

1.1.2. (unordered list): Listas não ordenadas, onde não tem importância de numeração e qual item vem depois, sendo marcadas por bolinhas, discos e quadrados.

1.2. Independente de qual lista usar, a tag precisa ser fechada.

1.3. Para criar itens dentro das listas, independentemente de qual esteja usando, precisa utilizar a tag <li> (list iten) que pode ou não ser fechada ao final do item ao qual se refere, nesse caso, o fechamento da tag fica ao seu critério.

1.4. Para criar uma estilização para as listas, nós precisamos classificar os itens. É como colocar um id=”nome”, mas funciona de melhor maneira para listas. O parâmetro (ou propriedade) que utilizamos é *class=”nome”*. Exatamente igual ao *id=”nome”*, só muda o parâmetro. No caso de listas precisamos colocar o mesmo nome para todos os itens, dessa forma, podemos alterar todos de uma mesma maneira.

1.5. Ao contrário do identificador, onde no css usamos a tralha (#), para nos referirmos ao item, no caso das classes, utilizaremos um ponto (.) e o nome dado para a classe. Ex.: .itens {} (referenciando uma classe em css) e #itens {} (referenciando um identificador em css).

1.6. Para deixar um elemento (ou uma classe deles) em itálico, podemos utilizar o marcador *font-style> italic;*.

1.7. Nós podemos trabalhar com divisões em uma página para podermos classificar diferentes conteúdos, para isso utilizamos a tag

1.7.1. Elas não interferem no visual ou no conteúdo da página, elas servem apenas para marcar o conteúdo e a partir do css fazer os efeitos desejados.

1.7.2. Como todas as div’s possuem o mesmo nome, podemos diferenciá-las criando id ou class’s para cada uma especificamente.

1.7.3. Identificador serve para itens únicos e as classes para itens que vão ser repetidos. Porém, se quisermos criar um layout robusto com todas as técnicas e quisermos replicar esse conteúdo no futuro, o ideal é criarmos classes.

1.7.4. Sempre que quisermos marcar um conteúdo visualmente, vamos criar uma classe para ele.

1.8. Conteúdo block, conteúdo inline e conteúdo inline block:

1.8.1. Block: Quando um texto ou elemento ocupa 100% do espaço da página, não permintindo que nada mais fique na sua lateral. É o que ocorre com as listas, parágrafos dentre outras tags.

1.8.2. Inline: É o que acontece com imagens. Elas não bloqueiam o resto da página, permitindo que mais conteúdo possa aparecer e ser exibido na mesma linha que ela está, porém, não permite que eu altere espaçamento lateral, interno, externo dentre outros.

1.8.3. Inline block: Quando a tag possui as duas características, ela vai bloquear tudo na sua lateral, mas é de um tamanho fixo, ou seja, eu posso alterar o tamanho do block, espaçamento interno e exerto, dentre outros, deixando assim, que mais conteúdo caiba na sua lateral.

1.8.4. Todas essas são características da marcação *display: ;*, desse modo, podendo ser alterado em css. Pode-se fazer uma marcação de display para as <ul> e colocar ele como qualquer uma das características anteriores, ou seja: display: block/inline/inline-block;

1.9. Alinhamento vertical:

1.9.1. Em css o alinhamento vertical sempre é na parte inferior do conteúdo, para alterar isso, basta colocar o *vertical-align: top;*, assim, o conteúdo vai ser alinhado na parte superior.

6. Finalizando a Página:

6.1. O cabeçalho de uma página possui a própria tag, que nada mais é que uma div, mas possuí sua semântica de cabeçalho. A tag é <header></header>.

6.2. O ideal é utilizarmos class para todas as estilizações, porque se por acaso quisermos alterar um título e substituir por outro, vai acabar com o código, ficar uma bagunça. Portanto, sempre que for fazer uma estilização, coloque class específica para aquilo.

6.3. É interessante colocar um padding-left em títulos que não forem centralizados na página, dessa forma, existe um respiro para ficar visualmente bonito.

6.4. O título é sempre a coisa de mais destaque e que aparece primeiro na página.

Parte 2 – Posicionamento, Listas e Navegação

1. Aula 1 – Criando Uma Nova Página:

1.1. Mesmo que seja uma imagem, a primeira coisa que aparece na sua página tem que colocar a tag <h1>, já que é a coisa mais importante dele.

1.2. Crie uma lista para colocar os links de uma página.

1.2.1. Geralmente uma lista não ordenada;

1.2.2. Use no cabeçalho, fica legal.

2. <u>Aula 2 – Navegando entre Páginas</u>:

2.1. Links, para poderem te direcionar para outra página, precisam de âncoras, que são os elementos clicáveis na sua página.

2.2. Para criar uma âncora para anexar um link, utilizamos a tag <*a*> (ancor), dentro dessa tag, é necessário que tenha alguns parâmetros/propriedades: *href=”endereço”*, *rel=”relação”*, *target=”alvo”*.

2.2.1. A tag <a> é de conteúdo, portanto, precisa ser fechada após a frase, palavra ou imagem que queira usar como âncora.

2.2.2. Href=”endereço”: Esse endereço é o caminho que vai te levar até a outra página, onde o outro arquivo .html está salvo, ou a um link externo.

2.2.3. Rel=”relação”: Na relação, precisa ser colocado qual é a relação desse link com sua página:

2.2.3.1. External: Para links externos, ou seja, que não são seus/do seu site, mas sim de outros.

2.2.3.2. Next: Para links de sua própria página, mas que levam para uma próxima.

2.2.3.3. Prev: Para links que voltam à página anterior a que você está agora.

2.2.4. Target=”alvo”: Nesse parâmetro é onde colocamos qual o alvo do nosso link.

2.2.4.1. _blank: Para links externos de outras páginas. Utilizando esse target, o link irá abrir uma nova guia em branco no navegador e abrir a página do link nela, ao invés de sobrepor seu site, dessa forma, mantendo o cliente no seu site por mais tempo.

2.2.4.2. _self: Para links internos, ou seja, cujo alvo seja sua própria página. Nesse caso, como o cliente continua no seu site ao clicar no link, não tem problema que ele seja carregado na guia atual em

que o cliente está, ou seja, não tem necessidade de abrir outra guia e carregar o link nela, acaba sendo desgastante se isso acontecer, pois fica uma bagunça de guias no navegador do cliente do seu site.

2.2.5. Geralmente o *taget* vem antes do *rel* na ordem dos parâmetros.

2.2.6. Caso a âncora que você esteja criando ainda não possua um link para colocar, no lugar do endereço no href, você deve colocar um "#".

2.3. Para criar um menu de navegação no nosso site, existe a tag **<*nav*>** (navegation). Devemos colocar a lista do menu de navegação que criamos, dentro dessa tag

2.3.1. Para que o menu de navegação não fique com os círculos ou discos por conta da lista, podemos alterar o display para inline no css: nav li {display: inline;}

2.4. Não é o ideal colocar um texto em caps para poder deixar tudo em letra maiúscula visualmente, isso vai contra as etiquetas da internet, uma vez que, quando um texto está em caps, significa que você está gritanto esse texto. Por conta disso, devemos escrever o texto em captilized padrão e depois alterar nos estilos para que ele fique normal no conteúdo, mas apareça em caps visualmente.

2.4.1. Para fazer essa alteração, podemos marcar apenas as âncoras no css (nav a {}) e colocar o *text-transform: uppercase;*. Dessa forma, todo o texto dos links ficam em maiúsculas.

2.5. Podemos deixar o texto em negrito alterando o peso da font para bold ou 900: *font-weight: bold/900;*.

2.6. Podemos retirar o sublinhado do texto das âncoras colocando o *text-decoration: none;*, ou seja, nenhuma decoração no texto.

2.7. Podemos colocar o espaçamento do item de 2 formas:

2.7.1. Especificando a direção na marcação, ou seja, *margin-top/left/right/bottom: ;*.

2.7.2. Colocando a forma geral e os valores do espaçamento correspondente para cada direção: *margin: 0 0 0 15px*, sendo eles: top, right, bottom e left, respectivamente.

3. <u>Aula 3 – Posicionamento dos Elementos</u>:

3.1. Existe um arquivo chamado reset.css que, literalmente, reseta todas as configurações css criadas automaticamente pelos navegadores.

3.2. Colocando a *position: relative;* em um elemento, nós podemos alterar ele da maneira que quisermos, posicionar para baixo, cima, qualquer lugar... A *position: static;* é a default, por tanto, você não consegue alterar nada das posições do seu elemento. Por isso que é necessário fazer essa alteração.

3.2.1. Uma vez que a position é relativa, nós podemos alterar ela simplesmente por colocar a direção que queremos e a quantidade em pixels: *top: 10px;/left: 50px;*...

3.2.2. Essas alterações são relativas ao ponto inicial daquele elemento. O ponto inicial dos elementos, geralmente é o cantinho da parte superior esquerda.

3.2.3. Se você alterar a posição, o elemento pode estar deslocado visualmente, mas seu ponto inicial continua no mesmo mesmo lugar, por conta disso, se você alterar a posição dele em um <header> e colocar bem para baixo, em algum momento ele vai ficar visualmente fora da box do <header>, onde nem mesmo a cor do background acompanha, deixando visivelmente claro.

3.2.4. O ideal é utilizarmos o posicionamento absoluto em relação a outra coisa, seja a página toda ou somente ao cabeçalho.

3.2.4.1. Quando coloca-se um elemento como posicionamento absoluto, em um cabeçalho, por exemplo, ele irá sair do cabeçalho, fazendo com que ele até diminua de tamanho. Isso ocorre porque a partir desse momento, esse elemento já não faz mais parte do cabeçalho. Ele saiu do lugar.

3.2.4.2. Com esse posicionamento, podemos colocar o elemente em absolutamente qualquer lugar que quisermos na tela, basta utilizar as marcações de posicionamento: *top: 0px;* (elemento cola na parte superior da página, pois ele tem 0 de distância entre ele e o topo).

3.2.5. Quando coloca absoluto em posicionamento de um elemento, ele deixa de fazer parte de tudo e fica posicionado em um ponto escolhido com relação à caixa.

3.2.5.1. Se quiser que esse elemento fique dentro de uma box, é necessário que essa box levante, desse modo, o seu elemento irá ter um posicionamento absoluto relativo à caixa.

3.2.5.2. Já dei a dica aqui acima... Se quiser que o elemento tenha um posicionamento absoluto, mas relativo à box, a box precisa ter um posicionamento relativo, pois, desse modo, ela irá levantar, alcançando o elemento e fazendo com que ele volte a fazer parte dela.

3.3. É interessante utilizarmos box com width de 940px, pois está dentro do ideal.

3.3.1. Para que uma box de cabeçalho fique sempre no centro, o ideal é deixar o top em 0 e o left/right em auto (*width: 0 auto;*), pois, desse modo, o cabeçalho vai ficar totalmente sem margem no topo e irá calcular automaticamente nas laterais, assim, dependendo do aparelho ou do tamanho da janela em que se encontra, a box sempre ficará centralizada horizontalmente.

3.4. Podemos seguir a mesma lógica do margin para configuração de distância do padding, ou seja, se tiver 2 configs de distância, a primeira será relacionada com top e bottom e a segunda com left e right, mas, se tiver 4 configs de diatância, será top, left, bottom e right.

3.4.1. Se tiver dúvidas, basta subir e rever como o margin é feito.

4. **Aula 4 – A tag *Section*:**
   4.1. Para a criação do conteúdo principal, nós podemos utilizar uma sessão <*div*>, porém, assim como no cabeçalho, agora no HTML5, nós possuímos uma tag mais semântica para alocar a criação do nosso conteúdo principal. A tag que utilizamos para isso é a <*main*>.
   4.2. Dentro de listas complexas podemos colocar títulos, imagens e parágrafos. Cartões de vários itens, vários links de um menu, dentre outras coisas, nada mais são do que listas que criamos e complexamos para dar um bom visual e ter um bom conteúdo.
   4.3. Vertical-align: local;: Carrega o conteúdo de acordo com a localidade específica. Se colocar em top, carrega de cima, left esquerda, etc.
   4.4. Trabalhamos o tempo todo com porcentagem e pixels, porém, isso pode acabar dando conflitos, pois se você está fazendo um cálculo de porcentagem da largura de vários elementos para que caibam dentro de uma box corretamente e por ventura acabar colocando um padding com valores de pixels, ele vai quebrar o seu design.
      4.4.1. Podemos concertar isso usando a marcação *box-sizing: border-box;*. Essa marcação fará com que todo o conteúdo de pixels colocado seja contado dentro da porcentagem utilizada na margem, que é o espaçamento externo.

5. **Aula 5 – Lidando com Bordas:**
   5.1. A borda tem 3 aspectos super importantes:
      5.1.1. **Tamanho: *border-width*;**
         5.1.1.1. Medido em px ou qualquer outra unidade de tamanho.
      5.1.2. **Tipo: *border-style*;**
         5.1.2.1. *Solid*: O mais usado, deixa bem forte e visível.;
         5.1.2.2. *Dashed*: Tracejado, deixa a borda toda pontilhada;
         5.1.2.3. *Doted:* Pontilhado;
      5.1.3. **Cor: *border-color*;**
         5.1.3.1. Segue a mesma lógica de coloração de todo o resto:
            5.1.3.1.1. #000000 = black = rgb(0,0,0)
            5.1.3.1.2. #ffffff = White = rgb(255,255,255)
            5.1.3.1.3. E assim por diante...
      5.1.4. Esses são os elementos que montam uma borda e suas configurações.
   5.2. Também podemos colocar as bordas especificamente em locais que quisermos, utilizando o mesmo princípio da margin:
      5.2.1. **Border-top;**
      5.2.2. **Border-bottom;**
      5.2.3. **Border-left;**
      5.2.4. **Border-right.**
      5.2.5. **Conjuntamente com isso, podemos configurar cada elemento específico de cada borda:**
         5.2.5.1. Border-top-width;

5.2.5.2.Border-top-color;
5.2.5.3.Border-top-style;
5.2.5.4. E assim subsequentemente para todos os outros também.

5.2.6. Porém, é muito chato ficar declarando cada uma delas independente toda vez, portanto, existe o modo universal onde você primeiro declara o tamanho, o tipo e depois a cor: ***border: 2px solid #000000;*** e isso irá substituir todas as outras 3 declarações.

5.3.Podemos arredondar as bordas utilizando o *border-radius: valorpx;*

5.3.1. Podemos utilizar essa declaração universal que irá arredondar todos os cantos, ou específicas, para arredondar somento um canto ou alguns cantos específicos:

5.3.1.1. Border-radius-top-left: arredonda o canto superior esquerdo
5.3.1.2.Esse mesmo princípio é seguido pelos outros cantos.

5.3.2. Também podemos colocar 4 valores diferentes para que ele arredonde cada canto em um tamanho específico.

6. Aula 6 – Pseudo-Classes CSS:

6.1. Para mapear quando o mouse está por cima de um elemento utilizamos a função *hover*.

6.1.1. Nós devemos colocar o elemento que queremos que seja mapeado e depois o hover: *a:hover{configuração diferente da já existente}*. No caso de links.

6.1.2. Como tem que ser uma configuração diferente, nós precisamos criar uma nova configuração para aquela tag que já existe: *nav a:hover {color: #C78C19}*, em caso de links da sessão de navegação, por exemplo.

6.2.Além do *hover* que serve para mapear quando o mouse está por cima do elemento, também podemos mapear quando o elemento foi clicado utilizando a função *active*. Segue o mesmo princípio do *hover*.

6.3. Se quiser fazer alterações de elementos que estejam dentro do elemento em que já está sendo afetado, na frente da função, seja ela a *hover*, *active* ou qualquer outra, é necessário declarar o elemento contido dentro desse primeiro que quer fazer a alteração, só aí ele irá mudar.

6.3.1. Porém, existe a necessidade de cirar outra configuração. Exemplo no ex da aula 6 do módulo/parte 2 do curso de HTML na Alura.

6.3.2.

7. Aula 7 – Finalizando a Página de Produtos:

7.1. O rodapé também possui uma tag específica, assim como as outras 2 sessões. Para criar um rodapé, utilizamos a tag <*footer*>.

7.2. Nós podemos colocar uma imagem de background na nossa página utilizando o css.

7.2.1. Utilizando a configuração *background: url("endereço");* nós podemos colocar a localização da imagem em endereço, desse modo, ao carregar a página, o background será a imagem que está salva lá.

7.2.2. Mesmo que a imagem utilizada seja pequena, quando utiliza essa configuração, o css vai copiar e colar essa imagem infinitas vezes até preencher todo o background daquele elemento.

7.3. Para adicionar caracteres específicos, como aquele c de copyright, por exemplo, precisamos saber o nome específico deles. Nesse site tem todos os carcteres de diversas línguas para podermos usar: https://unicode-table.com/pt/ .

7.3.1. O Unicode (conteúdo do site acima) é uma linguagem universal para representarmos símbolos.

7.3.2. O recomendado é utilizar a forma de "entidade" do símbolo, pois é o mais usado e abrange todos os navegadores.

Parte 3 – Trabalhando com Formulários e Tabelas

1. **Aula 1 – Criando uma Nova Página:**

1.1. Nós podemos criar formulários sempre que quisermos requerir informações do usuário.

2. **Aula 2 – Começando um Formulário:**

2.1. Para criar um formulário utilizamos a tag *<form></form>*.

2.1.1. Como dentro de um formulário o usuário coloca informações, ele está fazendo um *<input></input>*, e essa é a tag que usamos para receber essas informações.

2.1.1.1. Os inputs tem *type="tipo"* que precisam ser declarados, ou seja, se ele for colocar um nome, por exemplo, o *<input type="text">*.

2.1.1.2. Inputs são únicos, eles sempre precisam ser identificados com o id="nome", para que nunca sejam confundidos.

2.1.2. Dentro do *<form>*, tem outra tag sempre entra e anda junto com a *<input>* essa é a: *<label>*. Essa é a tag que se refere ao *<input>* e tudo o que é escrito dentro, aparecerá ao redor da caixa e de input servindo de guia para o que se deve escrever para preencher ele.

2.1.2.1. A label possui um parâmetro essencial: *for="id input"* onde você coloca o id do input ao qual ele é referente, dessa forma eles sempre andam juntos.

2.1.3. Fazer essa conexão entre o label e o input é super importante pois facilita a vida dos nossos clientes. Ex.: Se sua caixa de texto for muito pequeno ou quando temos muitas informações para serem entradas, basta o cliente clicar no nome que foi colocado no label e o cursos do teclado será direcionado para o input, não precisando clicar na caixa de texto certinha para isso.

2.2. Todo *<form>* possui um *<input>* final do *type="button"* e geralmente com o *value="enviar"*. Isto é porque é a partir deste botão que iremos mandar

todas as infos de preenchimento para o banco de dados, ou para onde quer que eles vão.

2.3. Todos os campos de um formulário possuem o *display: inline;* por padrão, portanto, os elementos ocuparão somente o tamanho de seus próprios conteúdos.

3. Aula 3 – Tipos de Campos Diferente:

3.1. Existem outros tipos de caixas de inserção de texto. No caso do formulário, onde as pessoas irão digitar somente uma linha, deve-se utilizar a tag <*input*>, mas, quando se quer criar uma caixa de texto onde o cliente pode digitar mais de uma linha, nós utilizamos a tag <*text-area*></*text-area*>, desse modo permitindo que o cliente te mande uma mensagem juntamente ao formulário.

- 3.1.1. Diferente da tag de input essa é uma tag de conteúdo, então ela irá abrir e fechar.
- 3.1.2. Existem 3 parâmetros que devemos colocar nessa tag, sendo eles:
  - 3.1.2.1. *Name=""*:
  - 3.1.2.2.*Id=""*: Dá uma identificação para ele
  - 3.1.2.3.*Cols="número de colunas"*: Configura quantas colunas a caixa de texto vai ter.
  - 3.1.2.4.*Rows="número de linhas"*: Configura quantas linhas a caixa de texto vai ter
- 3.1.3. Ele também é acompanhado do label.

3.2. Caixas de seleção também são usados a partir da tag <*input*>, mas em *type="radio"*.

- 3.2.1. Esse tipo de input possui outras 3 propriedades além dessa, sendo elas:
  - 3.2.1.1. *Name="nome do radio"*: Coloca-se o mesmo nome para todos os inputs radio que pertencerem ao mesmo grupo, caso contrário, ele irá permitir que selecione mais de uma opção
  - 3.2.1.2. *Value="do que se trata aquele radio"*: coloca-se o nome referente aquela radio, por exemplo, se for um radio de telefone ou de email, você coloca esses nomes.
  - 3.2.1.3. *Id="identificador"*: Coloca-se o nome específico para cada um, desse modo podendo assimilar um label a eles.
  - 3.2.1.4.Dica: Sempre marque um deles como *checked* (apenas isso dentro da tag), desse modo, o cliente será obrigado a escolher um.

3.3. Dica: Nós podemos polpar tempo colocando inputs dentro de labels, assim não tem a necessidade de criar id's.

3.4. No CSS podemos separar itens por uma vírgula (ou criar uma classe) quando queremos adicionar a mesma configuração para mais de um elemento.

3.5. Ordem de prioridade de marcadores em CSS:

- 3.5.1. Marcadores mais específicos primeiro form p{color: blue;} >> p{color: red;} = cor azul.
- 3.5.2. Se temos 2 tags, temos o dobro da força do seleteor, por isso ele se sobressai.

3.5.3. Tag: Força 1
3.5.4. Classe: Força 10
3.5.5. Id: Força 100
3.5.6. Estilo CSS Inline: Força 1000. Nada substitui ele.
3.5.7. Portanto: Inline >> Id >> Classe >> tag.

3.6. Podemos criar um campo do tipo <*select*> para criar um seletor, onde podemos colocar <*opinion*> para que possamos escolher. Ex.: Campos para data de nascimento em criação de contar.

4. <u>Aula 4 – Melhorando a Semântica</u>:

4.1. A acessibilidade é super importante e, quando falamos de celulares, é ainda mais, uma vez que a maior parte do tempo estamos utilizando um. Para que nosso site esteja adaptado para os celulares, existe um site que pode auxiliar: mobileinputtypes.com

4.1.1. Nesse site, ele mostra qual é o resultado exato de um input html de um site em um celular.

4.2. Nós podemos utilizar esses *types=""* mostrados no site acima no nosso site, dessa forma, facilitando a vida do usuário mobile.

4.2.1. Mesmo com essas alterações feitas, o input do nosso site irá continuar funcionando normalmente no computador.

4.3. <*Input type=""*> mais usados e que utilizamos no nosso formulário do exercício:

4.3.1. Email: Para o input referente
4.3.2. Tel: Para telefone.
4.3.3. Quando utiliza esses types específicos para cada coisa, o input identifica o que está faltando e te avisa para colocar quando tenta enviar o formulário. Ex.: Se você tentar enviar o forms, mas com o e-mail faltando sua extensão (@domínio.com[.br]), o navegador irá avisar que está faltando essas coisas e não te permitirá enviar o formulário.

4.4. Quando queremos que um campo específico seja preenchido pelo usuário, podemos colocar uma palavra reservada: *required*, dentro da tag do <*input*>, desse modo, o site não permitirá o envio do formulário a menos que esteja preenchido.

4.4.1. Quando faz isso, ele mostra uma mensagem de aviso no campo em que falta preencher e fala o que precisa ser preenchido.
4.4.2. Isso facilita a vida do usuário e impede o envio de um formulário com informações faltando.

4.5. Nós podemos colocar sugestões de preenchimento dentro das caixas de input colocando a propriedade *placeholder="mensagem"*, desse modo, a mensagem que foi colocada dentro irá aparecer de forma mais opaca dentro da caixa, guiando e induzindo o usuário a colocar do jeito que está escrito.

4.5.1. Essa propriedade é colocada dentro da tag do <*input*>!

4.6. Quando queremos agrupar os campos em um formulário e ter um título para isso temos tags específicas que deixam nosso código melhor escrito, pois dá mais semântica a ele.
    4.6.1. <*fieldset*>: Divisão para vários campos referentes à configuração de um ou mais campos de um assunto específico. Podemos usar quando temos preenchimento de dados de um cartão de crédito, dados de endereço ou bolinhas de seleção, por exemplo.
        4.6.1.1. Dentro dessa tag não temos parágrafos, apenas títulos que são definidos com a tag <*legend*>.

5. Aula 5 – CSS Avançado:

5.1. Nós podemos colocar delay nas transições de coloração dos Elementos em CSS, quando utilizamos a função *:hover{}* para criar interatividade. A marcação para isso é a *transition: tempo(e o "s" do lado) e a característica do elemento (background por exemplo);*. Dessa forma o css faz com que o seu background seja transicionado dentro do tempo determinado.

5.2. Conseguimos alterar a forma do ponteiro do mouse para uma mãozinha para reforçar que aquele elemento onde o mouse está em cima é clicavel. Para isso usamos a configuração *.elemento:hover{cursor: pointer;}*.

5.3. Existe uma propriedade chamada *transforms* em css que nos permite fazer alterações no nosso elemento de maneira super rápida e prática.
    5.3.1. Um exemplo é quando você quer que o elemento ao qual o mouse está em cima, aumente proporcionalmente todas as suas características (como borda, fonte, largura, dentre outros) em 20%.
    5.3.2. Podemos fazer isso usando: *transform: scale(1.2);*. Exige uma pequena "conversão" para ser utilizado, mas é muito simples.
    5.3.3. Scale(2) = 100% do valor padrão. Scale(1.2) = 20% do valor padrão.
    5.3.4. Para que ele funcione corretamente com todas as características do elemento, na configuração da transition a característica tem que ser "all": *transition: 1s all;*. Desse modo, todas as características serão transicionadas em 20% dentro de 1 segundo quando o mouse estiver em cima do elemento.
    5.3.5. Outra propriedade do transform é poder rotacionar o elemento quantos graus você quiser: *transforms: rotate(70deg);*.
    5.3.6. Como no css tudo é lido de cima para baixo, se a rotação estiver abaixo da escala, então essa configuração irá sobrescrever a escala e o elemento irá apenas rotacionar.
    5.3.7. Para poder adicionar as duas configurações juntas e funcionais, é só colocar as duas opções dentro de um mesmo transform separadas por um espaço: *transform: rotate(70deg) scale(1.2);*. Desse modo, as duas irão rodar tranquilamente sem serem sobrescritas.

6. Aula 6 – Estruturas de Tabelas:

6.1. Apesar de ser trabalhoso de se fazer por ter que utilizar muitas tags, as tabelas são coisas muito simples de se fazer.

6.2. Para começar a criação de uma tabela usamos a tag <*table*>.

6.3. Uma tabela é feita de linhas e colunas, portanto, para criar as linhas usamos: <*tr*> (table row) e as colunas: <*td*>. Essa tag na verdade é usada para criar células, portanto, se a sua tabela tiver 2 células, você irá colocar 2 <*td*> (um abaixo do outro) dentro de cada <*tr*>. E é dentro das células que colocaremos o conteúdo.

6.4. Podemos deixar uma tabela ainda mais semântica colocando tags específicas para cada parte dela:

- 6.4.1. Cabeçalho: <*thead*>
- 6.4.2. Conteúdo: <*tbody*>
- 6.4.3. Rodapé: <*tfoot*>
- 6.4.4. Cada uma dessas tags irá apenas englobar a tabela já construída, ou seja, as tags acima. A única que irá mudar é a tag da célula. Ao invés de colocar <*td*> colocamos <*th*>, indicando que essa célula é do head da tabela e deixando ainda mais semântico.

6.5. As tabelas também nos oferecem a possibilidade de juntar células e montar um visual diferente. Por exemplo, quando uma linha, que deveria ter 5 células, passa a mostrar só "uma célula". Esse efeito é conseguido através da propriedade colspan=X, onde X é o número de células que você quer agrupar.

Parte 4 – Avançando no CSS

1. Aula 1 – Adaptando Página Inicial:

1.1. Quando nós temos uma div onde o conteúdo é um só, ou seja, tem seu título, parágrafo e depois fecha, como uma sessão, por exemplo, nós chamamos essa divisão de <*section*> ao invés de <*div*>.

- 1.1.1. Section é uma divisão que possui conteúdo complexo.

1.2. Sempre use um espaçamento proporcional ao tamanho da fonte para facilitar a forma de ler.

- 1.2.1. Seja ele espaçamento do título, entre parágrafos, etc...
- 1.2.2. Se o seu título for 2em, use espaçamento 1em para margin dele, dos parágrafos, dentre outros.

1.3. Se quisermos que o elemento (uma imagem, por exemplo) seja circundado pelo texto que está abaixo, podemos usar a propriedade *float: ;* em CSS3.

- 1.3.1. Ao utilizar essa propriedade, o elemento começa a flutuar na página, como se estivesse descolado dela utilizando o *display: absolute;*, mas, diferente do *display*, a sombra do elemento continua lá, o que faz com que o texto que esteja em volta se levante para acompanhar a imagem, mas não sobreponha o texto, dessa forma, o texto fica em volta do elemento imagem colocado.

1.3.2. Um problema de utilizar o float é que todos os elementos abaixo do item configurado com ele serão afetados, subindo junto na página.

1.3.3. Para limpar ele e criar uma barreira onde dali para baixo mais nada será afetado utilizamos a propriedade *clear: left;* caso o seu float tenha sido configurado como left, se não, coloque a configuração da propriedade clear igual a configuração da propriedade float e tudo será concertado.

2. Aula 2 – Conteúdo externo:

2.1. Uma fonte preparada para a web tende a funcionar melhor em todos os navegadores e ter um comportamento parecido em todos os SO's tendo um visual muito semelhante, tendendo a ficar igual. Essa é a diferença de uma fonte preparada para a web para uma que não é.

2.1.1. Existem as fontes proprietárias e as de código aberto, onde você precisa de autorização ou compra para utilizar, ou pode utilizar do jeito que ela é, ou você pode modificar e distribuí-la da maneira que quiser, respectivamente.

2.1.2. O site https://fonts.google.com/ possuí inúmeras fontes disponíveis de todos os tipos preparadas para a web para que você possa utilizar da maneira que preferir, baixando no seu pc para utilizar com uma ferramenta como fotoshop/gimp para testes visuais, ou você pode implementar o código dela no seu site e utilizar ela como sua fonte padrão.

2.2. Caso a gente queira, podemos adicionar mapas e outros conteúdos externos no nosso site.

2.2.1. Utilizamos para a tag <iframe>, mas não precisamos nos preocupar em digitar todo o link. Se o serviço disponibiliza essa função, como no caso do maps e youtube, você pode simplesmente pegar o link embed que será disponibilizado e nele já terá as tags, propriedades e tudo o que for necessário para que funcione, o seu trabalho é simplesmente colar ele no HTML do seu código.

3. Aula 3 – Melhorando o CSS:

3.1. Para usarmos o inline-block em listas e fazer com que ocupem só a porcentagem de largura que você determinou, ela precisa estar dentro de uma div e sem nenhum espaço depois do , caso contrário, o conteúdo que deveria vir na lateral não irá ocupar o espaço da lateral.

3.2. Podemos colocar espaçamento entre as linhas de um conteúdo utilizando a propriedade *line-height: unidade de tamanho;*. Lembre-se de sempre colocar proporcional ao tamanho da fonte.

3.3. Quando temos uma lista, ou vários elementos com uma mesma classe, podemos utilizar a pseudo classe *.classe:first-child{}* em css se quisermos fazer alterações somente no primeiro elemento dessas classes. Podemos selecionar também o último, marcando como *last-child*.

3.3.1. Para selecionar qualquer outro elemento da lista, podemos utilizar o *nth-child(n)*, e colocar o número do elemento. Começa no 1 mesmo.

3.3.2. Podemos, dentro na lógica anterior, colocar um número e mais o "n" do lado, indicando assim, que deve-se pegar somente os números pares, ou ímpares *nth-child(2n)*.

3.4. Podemos criar um background com degradê utilizando a propriedade *background: linear-gradient(posição [deg], cor1, cor2...);* desse modo, criando um gradiente com quantas cores quiser.

3.4.1. Também temos o *radial-gradient*.

3.4.2. Podemos escolher onde o degradê começa a partir de cada cor colocando %.

3.5. Utilizando pseudo classes podemos selecionar coisas bem específicas, inclusive uma única letra de uma grande frase usando o *.classe:first-letter{}* a última com *last-letter;* e as demais com

3.6. Podemos seguir a mesma lógica anterior para parágrafos usando o *.classe:first-line{}* e o *last-lane{}*.

3.7. Temos outros dois pseudo elementos que servem para colocar conteúdo utilizando css antes ao qual a classe é referida e depois dela, as propriedades são *.class:before{}* e a *.class:after{}*, respectivamente. Ex.: Se tiver um título, e colocar uma classe nele, você pode usar essas configs para adicionar colchetes entre o título, mas sem alterar seu conteúdo, apenas visualmente.

3.7.1. Utilizando essa lógica, nós podemos utilizar códigos Unicode para adicionar, por exemplo, estrelinhas antes dos itens de uma lista.

4. <u>Aula 4 – Selecionando Qualquer Coisa</u>:

4.1. Utilizando o ">" no CSS depois de uma tag mãe e antes de uma tag filha, eu digo que estou marcando todas as filhas diretas dessa tag mãe, desse modo, se houverem outras tags do mesmo tipo, mas que filhas de outras filhas da tag mãe, elas são serão selecionadas.

```
Ex.: <main>
            <p>bom dia</p>
            <section class="principal">
                <h2 class="titulo-
principal">Sobre a Barbearia Alura</h2>
                <img src="../imagens/utensilios.jpg" alt=
"utensilios utilizados por um barbeiro" class="utensilios
">

                <p>Localizada no coração da cidade a <str
ong>Barbearia Alura</strong> traz para o mercado o que  h
á de melhor para o seu cabelo e barba. Fundada em 2019, a
```

```
 Barbearia Alura já é destaque na cidade e conquista novo
s clientes a cada dia.</p>

                <p id="missao"><em>Nossa missão é: <stron
g>"Proporcionar auto-
estima e qualidade de vida aos clientes".</strong></em></
p>

                <p>Oferecemos profissionais experientes e
 antenados às mudanças no mundo da moda. O atendimento po
ssui padrão de excelência e agilidade, garantindo qualida
de e satisfação dos nossos clientes.</p>
            </section>
```

4.1.1. Utilizando essa técnica (*main > p{ config }*), eu selecionaria somente o conteúdo dentro da * Bom dia *. Nenhuma outra * * seria afetada.

4.1.2. Seguindo a lógica, eu posso selecionar somente o primeiro parágrafo que vem após uma imagem utilizando *img + p {}*, ou, para selecionar todos os parágrafos após uma imagem usa-se *img ~ p {}*.

4.1.3. Podemos também, selecionar todos os itens MENOS um específico, utilizando ID no parágrafo, por exemplo. Usamos *.principal p:not(#missão){}* ainda utilizando o exemplo do código acima. Dessa forma, todas as configs dentro dos {} serão aplicadas a todos os menos o que tem *id=”missão”*.

4.2. Nós podemos calcular qualquer coisa com CSS.

4.2.1. Dentro de uma propriedade podemos utilizar a configuração *calc(operação)* e o css vai fazer as contas pra gente. Ex.: width: calc(40% - ((26px * 4) /4)); e ele irá nos mostrar o resultado preciso no tamanho do elemento que está sendo configurado.

5. Aula 5 – Opacidade e Sombra:

5.1. Opacidade é uma camada em cima do elemento que ajuda a luz a passar ou não, como um insulfilme.

5.2. Usamos a propriedade *opacity: 0-1(0.1;0.2...);* para configurar se irá ver a imagem claramente ou não. Geralmente usada com a função *.class:hover{}* e a *transition: ns;*.

5.3. Podemos usar opacidade em tudo: Elementos todos e cores.

5.4. Em cores, hoje em dia temos a config *alfa*, em RBG -> RGBA, podendo acrescentar mais um valor para determinar a opacidade. RGB(0,0,0) = RGBA(0,0,0,0)

5.5. Para adicionar sombras utilizamos o *box-shadow: npx(eixo x) npx(eixo y) npx (solido para transparente) npx(tamanho das bordas) color;*. Podemos colocar mais de uma na mesma box-shadow, basta separar por uma vírgula cada uma mais abaixo da outra.

5.5.1. Também podemos criar box-shaddow: inset npx npx npx color; para criar uma shadow interna ao elemento.

5.5.2. Podemos criar shadows em textos com text-shadow: ; O resto se segue igual.

6. <u>Aula 6 – Design responsivo</u>:

6.1. Podemos simular o site no celular usando a toolbar do chrome (ctrl+shift+i).

6.2. Para começar a adaptação para celular, nosso site precisa de outra tag <meta> no <head> com os seguintes parâmetros: *<meta name=”viewport” contente=”width=device-width”>*. Só de coloca essa tag o nosso site já começa a se comportar de forma diferente no modo celular.

6.3. Depois precisamos ir no final do nosso CSS e colocar uma media query para o nosso navegador, perguntando para ele em que tipo de tela estamos, celular ou pc.

6.3.1. @media screen and (max-width: 480px) {} é o que usamos para isso. Basicamente está dizendo que para telas até 480px ele irá exibir uma configuração de aparência diferente daqui configuramos até agr, adaptada para celular.

6.3.2. Quando estamos fazendo um site responsivo, nós iremos alterar no @media somente a característica/configuração do elemento, que queremos que se altere. Ex.: Se eu tiver feito toda a configuração do cabeçalho e para celular eu só quero que mude o width, eu não vou escrever tudo novamente, vou apenas reescrever o width.

7. <u>Aulas extras – Emmet:</u>

7.1. O vscode já vem com o Emmet integrado o que basicamente permite utilizarmos abreviações para agilizar o processo de coding. Ao invés de digitarmos tag por tag, colocando o conteúdo e classes de todas as tags um por um, nós podemos escrever por exemplo: main.conteiner>h1.tituloPrincipal{Este é o Título Principal}+p{Um bom dia a todos}+h2.tituloSecundario{Boa Tarde}... E assim por diante. O resultado após apertar enter é esse:

```
<main class="conteiner">
    <h1 class="tituloPrincipal">Este é o Título Principal</h1>
    <p>Um bom dia a todos</p>
    <h2 class="tituloSecundario">Boa Tarde</h2>
</main>
```

7.2. Os “.” Depois do nome de uma tag significa que você está adicionando uma classe para a tag, cujo o nome é o que vem asseguir.

7.3. Sinais de “>” representam que o que vier depois está em seguida na hierarquia, portanto, dentro uma divisão temos <h>’s, <p>’s, <img>’s dentre

outras tags e, colocando tudo isso depois do “>” significa que irá entrar na divisão principal. O mesmo ocorre se quiser colocar <h> dentro de <img> por exemplo.

7.4. Para indicar que dentro de uma divisão terão várias tags uma abaixo da outra, colocamos o “+” para indicar que estão no mesmo patamar hierárquico.

7.5. Os sinais de “{}” significa que o que estiver dentro é o conteúdo daquela tag.

7.6. Podemos também criar listas mais facilmente utilizando o “*” depois de um li seguido da quantidade de vezes que deseja que se repita.

7.7. Utilizando o “$” indicamos para o Emmet que queremos que ele conte e coloque o número de vezes que algo se repetiu. Ex.: ul.lista>li.conteudo${item $}*8... Aqui indicamos que queremos que seja repetido 8 vezes e o conteúdo dos itens será “item” mais o número correspondente daquela repetição. Resultado final:

```
<ul class="lista">
    <li class="conteudo1">item 1</li>
    <li class="conteudo2">item 2</li>
    <li class="conteudo3">item 3</li>
    <li class="conteudo4">item 4</li>
    <li class="conteudo5">item 5</li>
    <li class="conteudo6">item 6</li>
    <li class="conteudo7">item 7</li>
    <li class="conteudo8">item 8</li>
</ul>
```

7.8. Podemos também adicionar parâmetros às tags utilizando o sinal de [], portanto, no caso de uma imagem faríamos o seguinte: img[src=”img/imagem.png” alt=”imagem”]. Desse modo, a tag da img apareceria por completo com todos os parâmetros setados da maneira que você escreveu.

7.9. Um uso bem mais comum e que eu já faço diariamente é o de apenas escrever o conteúdo da tag, apertar enter e ela aparece apenas esperando colocar o conteúdo. Ex: “p+enter” para abrir e fechar a tag de parágrafo, dentre outros. Colocar o “!+enter” para criar toda a estrutura base do HTML5 dentre outras coisas.

7.10. Se colocarmos apenas um “.” Sem uma tag antes e o nome da classe em seguida, ele automaticamente cria uma div com a classe referente.

7.11. Conseguimos agrupar conteúdos específicos colocando “()”. Por exemplo, você escreve uma div toda com seu conteúdo e você quer criar outra div, mas embaixo daquela e não dentro, então para fazer isso, todo o conteúdo da primeira div, incluindo ela, tem que estar dentro de “()”, desse modo, você pode colocar um”+” ao lado do “)” e escrever a div que ficaria na parte de baixo.