Primeiros Elementos do Documento

O documento HTML5 precisa de alguns elementos básicos para funcionar. O primeiro deles é o <!DOCTYPE html>, que tem como função “avisar” para o navegador que o que vem a seguir é um documento html5.

Depois dissoo precisamos incluir o <html> (geralmente acompanhado do atributo lang), que vai envolver todo o código da página. Esse também é conhecido como o elemento raíz (mas tem menos “força de seleção” que o seletor “:root”, então os estilos atribuídos ao root sobrepõem os aplicados ao html, caso haja algum).

Depois temos o <head>, que é a área onde se encontram as configurações gerais do site.

Formatação Básica

O navegador não diferencia espaços ou quebras de linhas no código html, por isso todo uso de mais de um espaço ou Enter no código sempre vai resultar em apenas um espaço na visualização da página. Para criar um espaço em branco precisamos usar o código de caracter “&nbsp;”, enquanto pra quebrar uma linha precisamos usar a Tag <br/> (ou apenas <br>).

Em alguns casos é possível manter o mesmo espaçamento que está no código, isso quando usamos a tag “<pre>”. Tudo que está entre essa tag e oo fechamento dela vai ser mostrado com o exato espaçamento que tem dentro do código. Isso é útil para juntar com a tag <code> e ‘simular’ a visão de um código no navegador.

Além disso, o navegador sempre vai ler uma palavra inteira sem quebrar a linha. Para evitar isso, podemos marcar onde fica(m) a(s) separação(ões) de sílaba(s) da(s) palavra(s) com a tag “<wbr/>” (word break), funciona em links também. Essa tag permite que a linha quebre no ponto em que a tag marcar na palavra, porém essa quebra não vem com o hífen no final (como geralmente tem quando precisamos quebrar a linha no meio de uma palavra em nossa língua), mas também temos o código de caracter especial “&shy;” para permitir a quebra de linha com hífen no final da sílaba definida. Tanto a tag quanto o código precisam ser inseridos diretamente no ponto onde se quer que a quebra aconteça, caso venha a acontecer.

Podemos, também, marcar um trecho de conteúdo para ser mostrado acima da linha normal com a tag <sup>, ou abaixo da linha normal com <sub>.

Posicionamento Absoluto

O posicionamento absoluto é um posicionamento relacionado a um ponto do elemento pai do elemento em questão diferente do ponto inicial padrão.

Quando se usa o posicionamento absoluto, a posição dele fica definida em algum ponto específico no navegador, então isso pode resultar em um desalinhamento com as caixas de vez em quando, então sendo assim a melhor forma para se lidar com isso é inserindo uma caixa pai com posicionamento relativo. Isso ocorre porque a caixa precisa "levantar" o elemento posicionado para que ele faça parte dela.

É preciso tomar cuidado ao usar o "position: absolute;", pois o espaço que seria destinado a ele antes deixa de existir no documento.

Formulários

Quando uma página requer que o usuário digite alguma informação para realizar alguma ação, o local onde o usuário insere a informação geralmente é um formulário.

Os formulários têm como função coletar as informações inseridas pelo usuário para enviar para algum lugar (seja para página que vai tratar a informação de alguma forma, seja para um servidor que enviará as informações inseridas para o e-mail desejado, etc.).

Os formulários precisam ter dentro deles duas tags:

- A <input>, com o atributo tipo, para definir o tipo de input que deve ser inserida no campo desejado (Uma tag sem fechamento).

- A <label>, uma tag para explicar ao que o input se refere.

Um input também necessita de um atributo id para ligá-lo ao label. Isso acontece dando um id com o nome, por exemplo: “identificadorLabel” e inserindo no label um atributo “for” com o valor “identificadorLabel”. Exemplo:

    <main>

        <form>

            <label for="nomeSobrenome">Nome e Sobrenome:</label>

            <input type="text" id="nomeSobrenome">

        </form>

    </main>

- Isso facilita a seleção do campo de input em alguns casos, visto que basta clicar em cima do conteúdo dentro do label que o cursor vai parar dentro do campo onde será realizado o input.

- É possível ter mais de um par de layout + input no mesmo elemento form.

Dentro de todo form sempre deve ter também um input com tipo “submit” para enviar as informações coletadas para um lugar específico (seja um site, servidor para envio de email, etc...). Esse input em específico não necessita de uma label para fazer par. Exemplo:

        <form>

            <label for="nomeSobrenome">Nome e Sobrenome:</label>

            <input type="text" id="nomeSobrenome">

            <label for="email">E-Mail</label>

            <input type="text" id="email">

            <label for="telefone">Telefone</label>

            <input type="text" id="telefone">

            <input type="submit">

        </form>

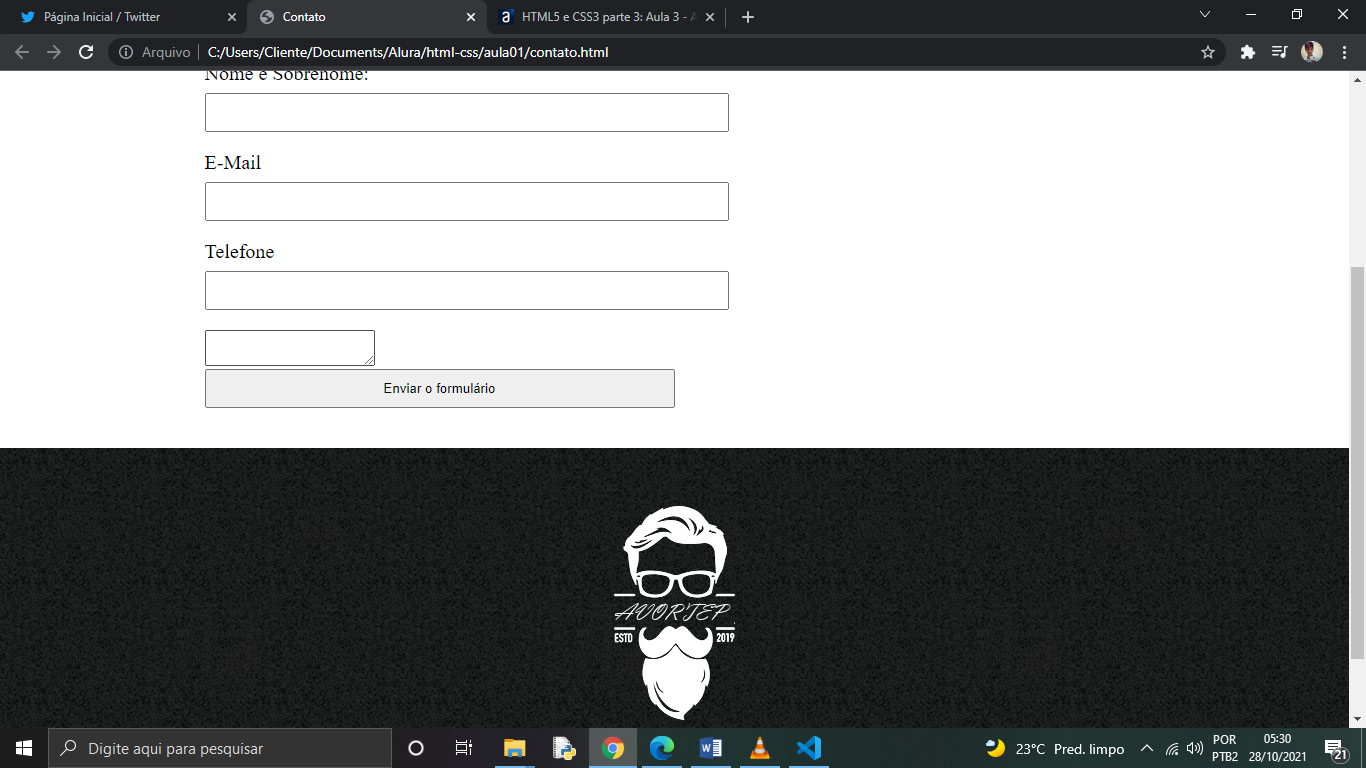
O que gera o seguinte resultado:  


- O “Enviar” que aparece é porque o navegador interpreta essa tag inserindo um botão com a palavra “Submit” no idioma especificado no início do documento html, porém é possível alterar ele inserindo o atributo “value”, que receberá o nome desejado para o botão como valor. Exemplo:

            <input type="submit" value="Enviar o formulário">

Que resulta em:  


Embora o input permita a entrada de texto, não é possível usar ele para suportar um campo de texto muito grande, e para isso temos o elemento textarea. Exemplo:



Como pode ser observado, a altura e largura do textarea é adaptável pelo próprio usuário, contudo também é possível especificar uma altura e largura padrão para que o textarea seja apresentado inicialmente da tela. Essas altura e largura na verdade são representadas por cols (colunas, o termo geralmente usado para definir o espaço que um caracter ocupa) e rows (linhas). Exemplo:

        <form>

            <label for="nomeSobrenome">Nome e Sobrenome:</label>

            <input type="text" id="nomeSobrenome">

            <label for="email">E-Mail</label>

            <input type="text" id="email">

            <label for="telefone">Telefone</label>

            <input type="text" id="telefone">

            <textarea cols="60" rows="12"></textarea>

            <input type="submit" value="Enviar o formulário">

        </form>

Cujo resultado foi:



(Naturalmente esses elementos apareceriam um do lado do outro, mas aqui foi feita uma alteração para que todos os input tivessem um display: block)

O elemento textarea também deve receber um label associado a ele, que pode ser feito como a associação do label e do input.

Além do text, outros tipos de input podem ser processados pela página, um deles é o input de marcar uma opção entre várias, também chamado de radio box. O radio box necessita de um value para explicar ao que ele é referente para o navegador (e ferramentas como Screen Reader, eu presumo).

Por padrão, o radio box não possui nenhum texto, mas pode ser adicionado um relacionado a ele assim como nos outros input – o elemento label. Exemplo:

            <label for="nomeSobrenome">Nome e Sobrenome:</label>

            <input type="text" id="nomeSobrenome">

            <label for="email">E-Mail</label>

            <input type="text" id="email">

            <label for="telefone">Telefone</label>

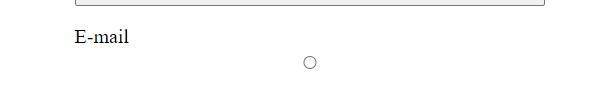
            <input type="text" id="telefone">

            <textarea cols="60" rows="12"></textarea>

            <input type="submit" value="Enviar o formulário">

            <label for="radioEmail">E-mail</label>

            <input type="radio" value="email" id="radioEmail">

Que resultaria na seguinte radio box: Como podemos ver, o texto e a radio box ficaram desalinhados, mas isso pode ser mudado via CSS.

Geralmente se deseja que um input do tipo radio permita a escolha de apenas um dos itens disponíveis, porém o input radio não faz isso por padrão, para isto precisamos adicionar o atributo name. Adicionando o atributo name para todas as opções do mesmo grupo indica para o navegador que aquelas opções são referentes à mesma pergunta, logo apenas uma delas pode ser selecionada. Exemplo:

        <form>

            <label for="nomeSobrenome">Nome e Sobrenome:</label>

            <input type="text" id="nomeSobrenome">

            <label for="email">E-Mail</label>

            <input type="text" id="email">

            <label for="telefone">Telefone</label>

            <input type="text" id="telefone">

            <textarea cols="60" rows="12"></textarea>

            <label for="radioEmail">E-mail</label>

            <input type="radio" name="contato" value="email" id="radioEmail">

            <label for="radioLigacao">Ligação</label>

            <input type="radio" name="contato" value="ligacao" id="radioLigacao">

            <label for="radioWhatsapp">WhatsApp</label>

            <input type="radio" name="contato" value="whatsapp" id="radioWhatsapp">

            <input type="submit" value="Enviar o formulário">

        </form>

Além da radio-box, existe também a checkbox, que geralmente tem como característica aceitar nenhuma, uma ou mais opções.

Além de linkar uma label ao input pelo id, também podemos poupar esse trabalho inserindo o input dentro do Label, que deve funcionar de forma semelhante. Exemplo:

            <div><!-- Essa div não tem nenhum grande propósito, é só pra indicar que isso tudo dentro dela pertence à mesma seleção de dados. É puramente organizacional. -->

                <p>Como prefere o nosso contato?</p>

                <label for="radioEmail">E-mail</label>

                <input type="radio" name="contato" value="email" id="radioEmail">

                <label for="radioLigacao">Ligação</label>

                <input type="radio" name="contato" value="ligacao" id="radioLigacao">

                <label for="radioWhatsapp">WhatsApp</label>

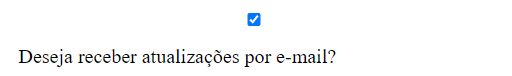
                <input type="radio" name="contato" value="whatsapp" id="radioWhatsapp">

            </div>

            <label><input type="checkbox">Deseja receber atualizações por e-mail?</label>

            <input type="submit" value="Enviar o formulário">

Que resulta em:



Para posicionar cada um desses cada um desses de maneira que queremos, devemos mexer no display e espaçamento deles, mas primeiro é necessário identificar os elementos nos quais a estilização é desejada para que o estilo não se aplique em lugares que não queremos. Exemplo:

<form>

            <label for="nomeSobrenome">Nome e Sobrenome:</label>

            <input type="text" id="nomeSobrenome" class="inputPadrao">

            <label for="email">E-Mail</label>

            <input type="text" id="email" class="inputPadrao">

            <label for="telefone">Telefone</label>

            <input type="text" id="telefone" class="inputPadrao">

            <label for="mensagem">Mensagem:</label>

            <textarea id="mensagem" cols="60" rows="12" class="inputPadrao"></textarea>

            <div><!-- Essa div não tem nenhum grande propósito, é só pra indicar que isso tudo dentro dela pertence à mesma seleção de dados. É puramente organizacional. -->

                <p>Como prefere o nosso contato?</p>

                <label for="radioEmail"><input type="radio" name="contato" value="email" id="radioEmail">E-mail</label>

                <label for="radioLigacao"><input type="radio" name="contato" value="ligacao" id="radioLigacao">Ligação</label>

                <label for="radioWhatsapp"><input type="radio" name="contato" value="whatsapp" id="radioWhatsapp">WhatsApp</label>

            </div>

            <label><input type="checkbox">Deseja receber atualizações por e-mail?</label>

            <input type="submit" value="Enviar o formulário">

        </form>

CSS:

form {

*margin*: 4rem 0;

}

form label, form p {

*display*: block;

*font-size*: 2rem;

*margin*: 0 0 1rem;

}

.inputPadrao {

*display*: block;

*margin*: 0 0 2rem;

*padding*: 1rem 2.5rem;

*width*: 50%;

}

Nos forms também veremos o elemento selection, que basicamente indica para o navegador que ali haverá uma barra com drop down com opções, que também são representadas por um elemento (option) que é aninhado dentro do selection. Exemplo:

            <div>

<p>Em qual horário prefere ser atendido?</p>

                <select>

                    <option>Manhã</option>

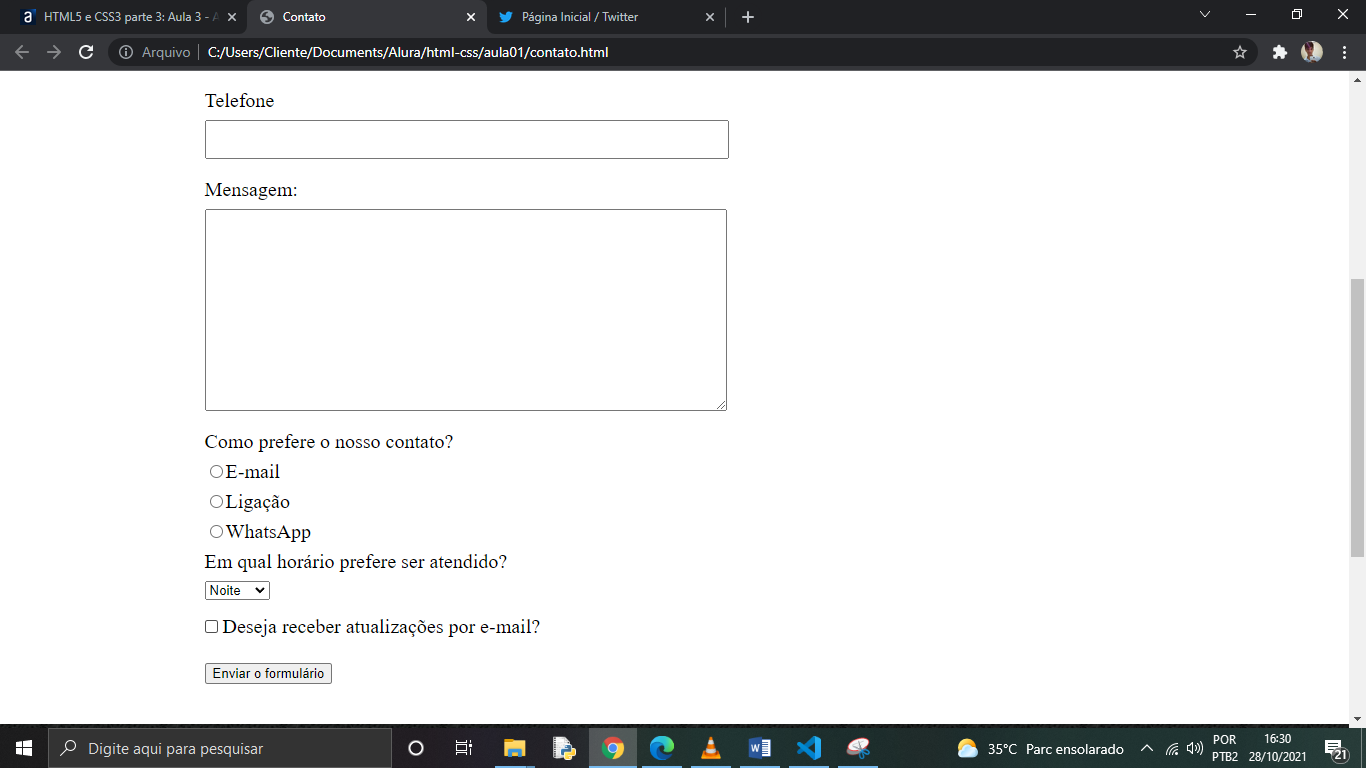
                    <option>Tarde</option>

                    <option>Noite</option>

                </select>

            </div>

Que gera o resultado:



Com isso em mente, agora podemos ver alguns ajustes que podem ser feitos para aprimorar a experiência de um usuário que esteja **acessando o site por um celular**.

Atributos de Input e Seus Efeitos nos Celulares

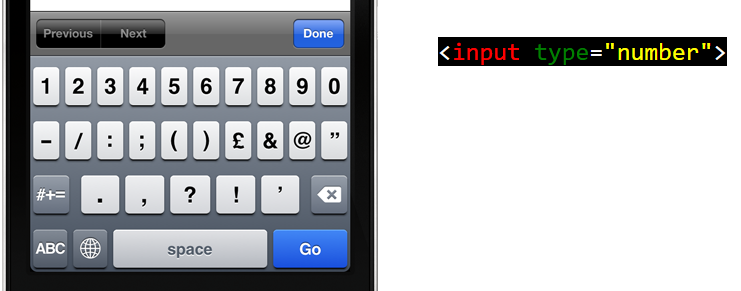
O input de tipo “text” no celular funciona por padrão com o teclado comum do celular. Isso é de se esperar para a maioria dos campos, mas no campo de email é possível colocar o tipo de input “email”, e assim o celular responderá apresentando o teclado com um “@” e um “.” ou “.com” já na face inicial do teclado. Exemplo:



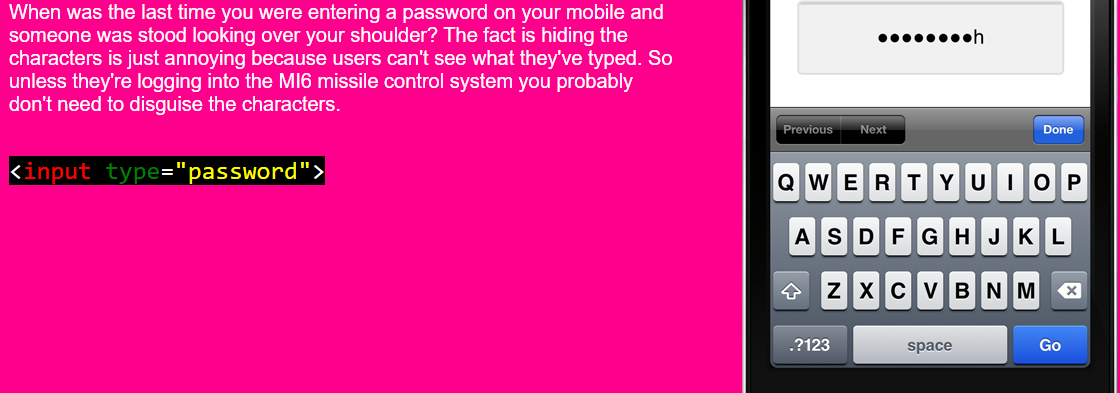
Para fins semelhantes, existe o input do tipo “tel”, que faz aparecer um teclado numérico quando clicado.



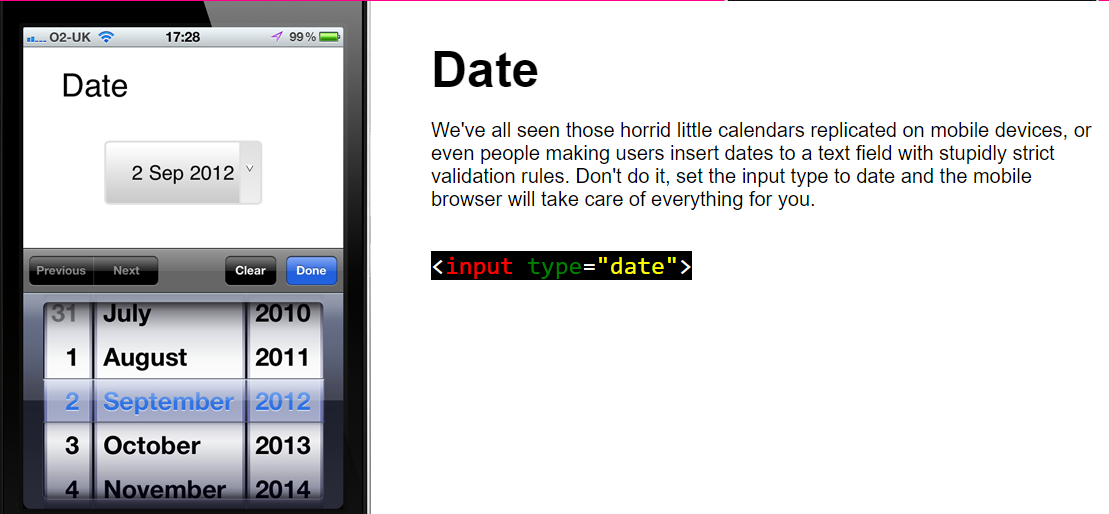
Esse tipo frequentemente é confundido com o tipo “number”, que apresenta um teclado numérico com símbolos de início.



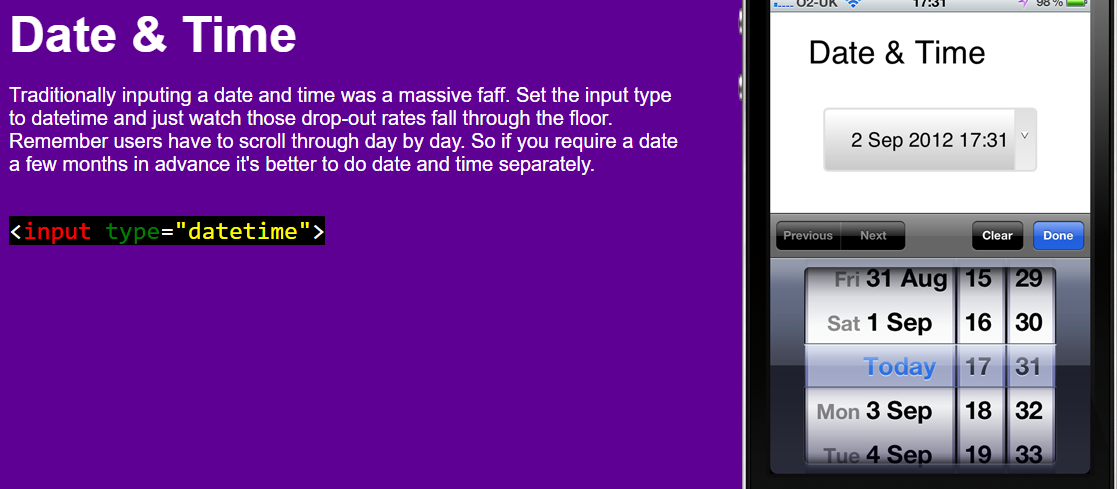
Também temos o tipo “password”, que esconde os caracteres no campo do input.



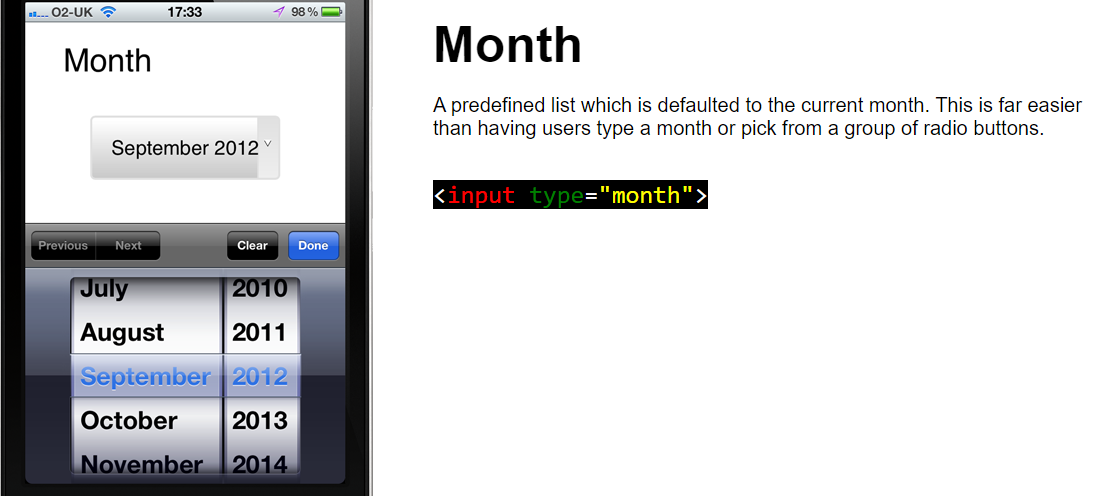
Podemos observar também o tipo “date”, que nos permite abrir uma input que oferece opções de data para o usuário.



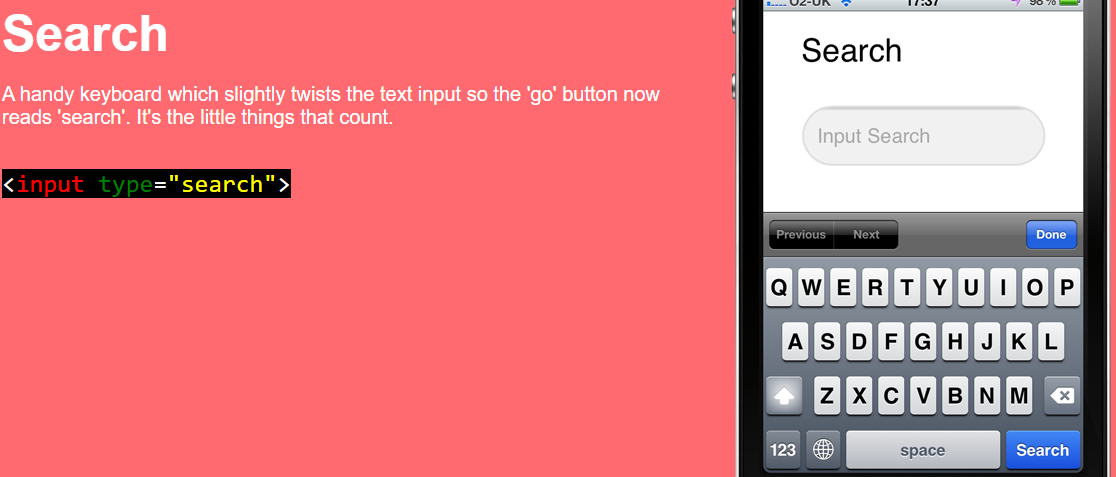
Além da data, também podemos especificar ainda mais com um input do tipo “datetime” para permitir nosso usuário escolher um horário e uma data.



Caso não queiramos ser tão específicos, tem o tipo “month”, que permite que o usuário escolha um mês.



E enfim temos o tipo “search” e no mobile ele basicamente substitui o botão de “Enter” ou “Go” por “Search” ou uma lupa.



Mais sobre Inputs e Preenchimento

SEMPRE que temos um input e queremos que ele seja preenchido obrigatoriamente para que o formulário seja efetivado, precisamos adicionar o atributo “required” nesse input. Exemplo:

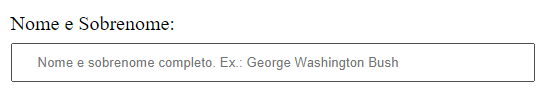
     <label for="nomeSobrenome">Nome e Sobrenome:</label>

     <input type="text" id="nomeSobrenome" class="inputPadrao" required>

Com isso, o navegador vai voltar àquele campo e indicar para o usuário preenche-lo caso ele tente enviar o formulário incompleto.

Ao usar os valores certos no atributo type temos ajuda do navegador não só para o celular. Nesse caso, se digitarmos um texto sem um “@” incluído no input de email, o navegador já identifica que o texto não se trata de um email e pede para o usuário corrigir.

Às vezes vemos nos inputs um exemplo padrão do tipo de input que deve ser inserido ali. Exemplo:



Isso se chama de “placeholder”, e este também é o nome do atributo para adicioná-lo ao nosso input. Exemplo:

      <label for="nomeSobrenome">Nome e Sobrenome:</label>

      <input type="text" id="nomeSobrenome" class="inputPadrao" required placeholder="Nome e sobrenome completo. Ex.: George Washington Bush">

Podemos fazer algo semelhante com o check box ou radio box com o atributo checked, para fazer com que uma das opções já apareça marcada inicialmente por padrão (no caso, o atributo deve ser usado apenas na opção aparecerá marcada). Exemplo:

        <form>

            <label for="nomeSobrenome">Nome e Sobrenome:</label>

            <input type="text" id="nomeSobrenome" class="inputPadrao" required placeholder="Nome e sobrenome completo. Ex.: George Washington Bush">

            <label for="email">E-Mail:</label>

            <input type="email" id="email" class="inputPadrao" required placeholder="seuemail@dominio.com">

            <label for="telefone">Telefone:</label>

            <input type="tel" id="telefone" class="inputPadrao" required placeholder="(XX)XXXX-XXXX">

            <label for="mensagem">Mensagem:</label>

            <textarea id="mensagem" cols="60" rows="12" class="inputPadrao" required></textarea>

            <div><!-- Essa div não tem nenhum grande propósito, é só pra indicar que isso tudo dentro dela pertence à mesma seleção de dados. É puramente organizacional. -->

                <p>Como prefere o nosso contato?</p>

                <label for="radioEmail"><input type="radio" name="contato" value="email" id="radioEmail">E-mail</label>

                <label for="radioLigacao"><input type="radio" name="contato" value="ligacao" id="radioLigacao">Ligação</label>

                <label for="radioWhatsapp"><input type="radio" name="contato" value="whatsapp" id="radioWhatsapp" checked>WhatsApp</label>

            </div>

            <div> <!-- De novo, uma div apenas para separar um grupo de opções já que esse elemento tem suas semelhanças com o radio -->

                <p>Em qual horário prefere ser atendido?</p>

                <select>

                    <option>Manhã</option>

                    <option>Tarde</option>

                    <option>Noite</option>

                </select>

            </div>

            <label><input type="checkbox" class="checkbox" checked> Deseja receber atualizações por e-mail?</label>

            <input type="submit" value="Enviar o formulário">

        </form>

Mais Sobre a Semântica dos Forms

Como podemos ver, nesse formulário foram inseridas duas divs para ajudar a organizar o conteúdo de múltipla escolha, porém no HTML5 há tags semânticas para cumprirem esse papel.

A primeira delas é a <fieldset>. Essa é a tag certa para envolver conteúdo referente ao mesmo grupo de conteúdo de um formulário (isso se aplica a perguntas, opções de barra com dropdown, campos de preenchimento referentes ao mesmo assunto com os de dados de um cartão ou um endereço).

Além disso, ao invés de usarmos o parágrafo, devemos usar o elemento legend (às vezes referido como field legend, mas a tag é <legend> mesmo). Este elemento é o que deve indicar o título/texto que precede uma séries de opções para escolha, e deve ser usado dentro do elemento fieldset.

Um Pouco Sobre a Propriedade Transition

Quando aplicamos um estilo ao :hover de um elemento, esse estilo dita como um elemento vai ficar quando passarmos o mouse por cima. Esse processo de aplicar as alterações baseadas no estado chama-se de transição.

A propriedade transition no css é na verdade um shorthand (abreviação) para as propriedades transition-property, transition-duration, transition-timing-function e transition-delay. Ela permite definir a transição entre 2 estados do elemento. Estados diferentes podem ser definidos usando pseudo-classes como :hover ou :active, ou dinamicamente (usando o JavaScript).

O valor padrão para cada propriedade transition é:

.botaoEnviar:hover {

*transition-delay*: 0s;

*transition-duration*: 0s;

*transition-property*: all;

*transition-timing-function*: ease;

}

Onde o **transition-delay** determina o tempo de duração de espera antes que o efeito da transição comece a acontecer.

O **transition-duration** tem como função determinar o tempo pelo qual o efeito de transição durará.

O **transition-property** dita/define a quais propriedades CSS a transição deve se aplicar.

O **transition-timing-function** define a aceleração da animação de transição durante seu tempo determinado.

Transformações

Atualmente o CSS também nos permite transformar elementos por meio de seu estado. Essa transformação pode ser, por exemplo, uma mudança de tamanho do elemento em questão usando o valor scale (proporção que deseja aumentar em relação ao tamanho atual). No exemplo aumentaremos 20% do tamanho do elemento com o transform:

.botaoEnviar {

*width*: 40%;

*padding*: 1.5rem 0;

*background-color*: orange;

*color*: white;

*font-weight*: bold;

*font-size*: 1.8rem;

*border*: none;

*border-radius*: 0.5rem;

*transition*: 1s all;

*cursor*: pointer; /\* dessa maneira fazemos o cursor mudar para o ícone de mão apontando \*/

}

.botaoEnviar:hover {

*background*: darkorange;

*transform*: scale(1.2);

}

Isto é possível pois o CSS muda as coordenadas no espaço da tela sem afetar o fluxo de informação da página. Essas alterações incluem também rotações, inclinações, tradução tanto em 2d quanto em 3d (além do aumento de escala mostrado anteriormente).

Duas propriedades são usadas para definir o CSS transforms: **transform** e **transform-origin**.

A propriedade **transform-origin** define a posição de origem do elemento. Por padrão essa posição é o canto superior esquerdo do elemento e pode ser movido. É utilizada para muitas alterações, como rotação, alteração de escala e inclinação, que precisam de um ponto específico como parâmetro.

A propriedade **transform** define que ação de transformação será realizada, e também pode aceitar mais de uma transformação, basta separar cada uma com blankspace. Exemplo:

Ao usar o seguinte código:

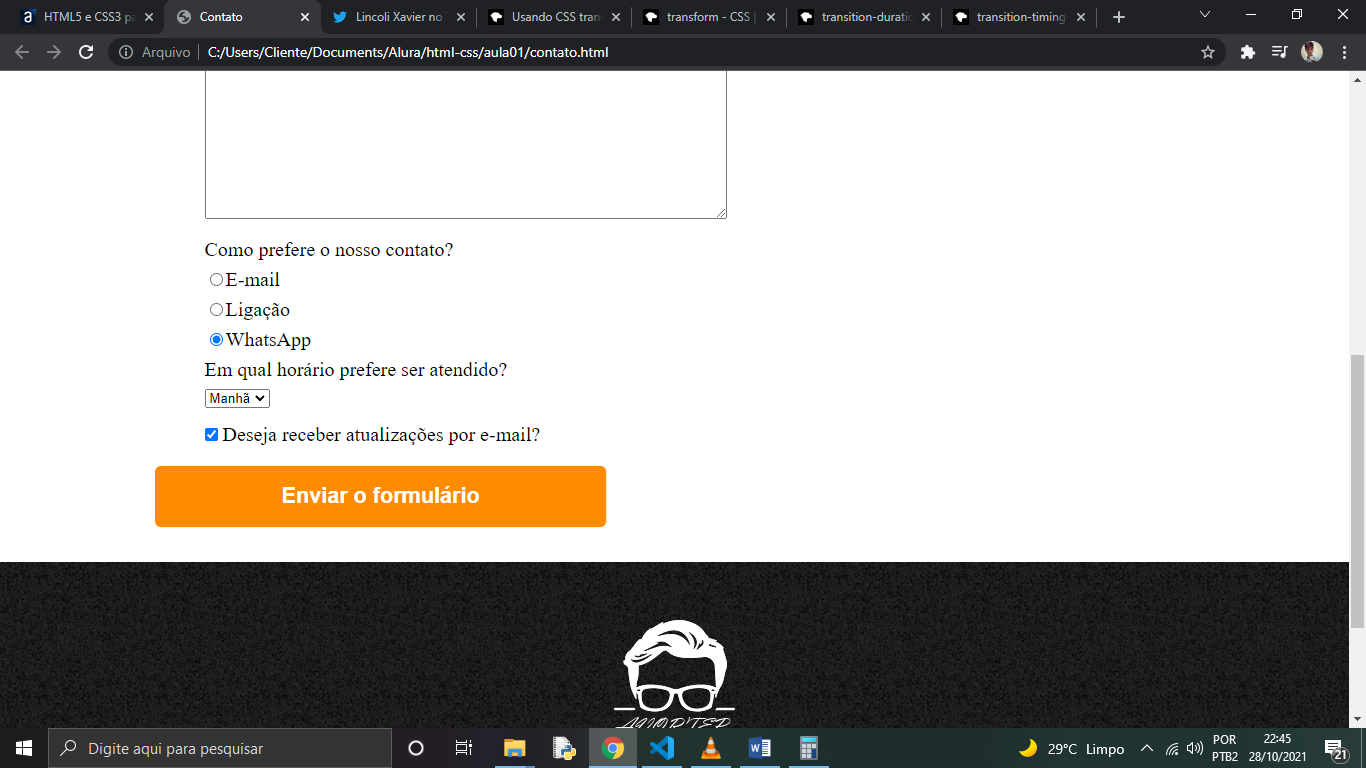
.botaoEnviar:hover {

*background*: darkorange;

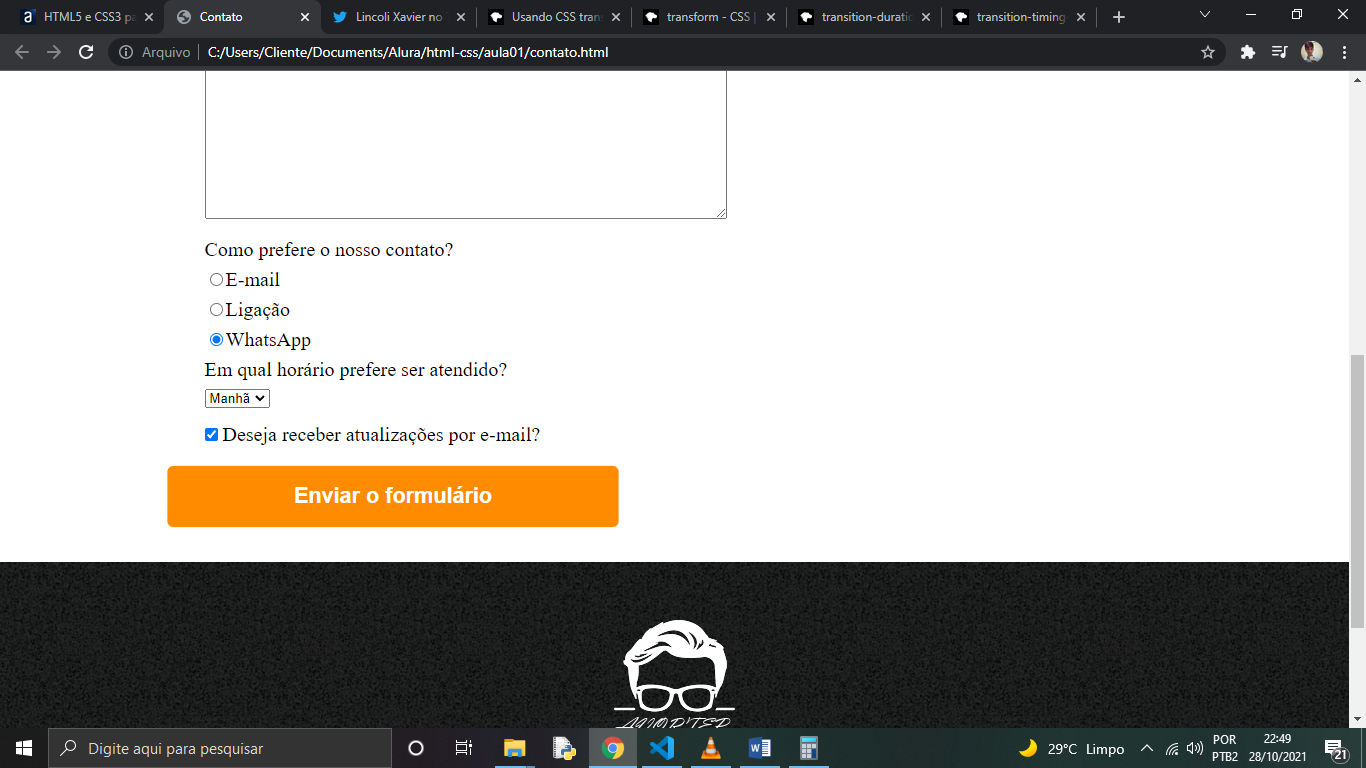
*transform*: scale(1.2);

*transform-origin*: 25rem;

}

Temos o seguinte resultado no elemento transformado:  


Isso se deve por causa do transform-origin que foi especificado como 250px (o tamanho da fonte no :root está como 62.5%, logo 25rem = 250px na tela do Desktop.



Como podemos ver, com o transform-origin definido, sua localização na transformação ficou diferente.

Tabelas

O elemento table indica para o navegador que vai ter uma tabela naquela área. As tabelas são compostas de linhas (table rows/tr) e células, podendo ser células de dados (table data cell/td) e células de título/cabeçalho (table header cell/th). No nosso código, primeiro especificamos quantas linhas terá a tabela, e depois quantas colunas teremos dentro de cada linha.

Geralmente a primeira linha possui uma descrição/informação para/sobre os outros itens que estão na mesma coluna, como um título ou gênero. O nome do elemento para especificar que uma área é a área de cabeçalho/título é thead. Ele precisa ser inserido aninhado dentro de um <tr> para ocupar uma linha da tabela é dentro dele que a tag <th> deve estar.

Além disso, também temos o elemento tbody, para definir qual conteúdo pertence ao corpo da tabela. Exemplo:

        <table>

            <thead>

                <tr>

                    <th>Dia</th>

                    <th>Horário</th>

                </tr>

            </thead>

            <tbody>

                <tr>

                    <td>Segunda</td>

                    <td>08:00-18:00</td>

                </tr>

                <tr>

                    <td>Terça</td>

                    <td>08:00-20:00</td>

                </tr>

                <tr>

                    <td>Quarta</td>

                    <td>08:00-20:00</td>

                </tr>

            </tbody>

        </table>

Com a estilização:

table {

*margin*: 2rem 0 4rem;

}

thead {

*background*: #555555;

*color*: white;

*font-weight*: bold;

}

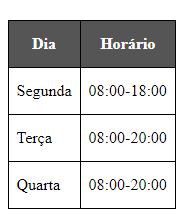
td, th {

*border*: 1px solid black;

*padding*: 1.5rem 0.8rem;

}

O resultado fica:

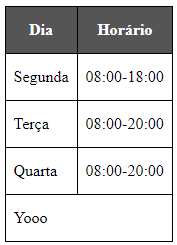


Além disso, podemos fazer uma célula ocupar o espaço de mais de uma coluna na tabela, basta inserir o atributo “colspan” com o valor numérico da quantidade de colunas se deseja que aquele elemento ocupe. Exemplo:

                <tr>

                    <td colspan="2">Yooo</td>

                </tr>



Só é possível colocar um valor de colspan igual ou menor à quantidade de colunas já presentes na tabela.

Incorporando Conteúdo Externo às Nossas Páginas

Frequentemente vemos páginas com áreas dedicadas à apresentação de um vídeo do youtube ou um mini mapa apontado para uma localização específica no google maps. Esse tipo de conteúdo é representado pelo elemento iframe.

O iframe tem como função simular uma janela com largura e altura específicas que pode reproduzir conteúdo externo. Por ter essa altura e largura específica, o processo para tornar ela responsiva a uma variedade de telas é diferente do adotado para tornar imagens ou vídeos internos responsivos.

A ideia por trás disso envolve criar uma div ao redor do iframe -> dar posicionamento absoluto ao iframe e indicar suas coordenadas -> aumentar o padding bottom da div contentora do iframe (isso serve para que a div ocupe um espaço maior. Como o posicionamento absoluto tira o elemento do seu posicionamento original e basicamente anula o espaço que ele ocupava, é preciso fazer isso para que ele não fique flutuando por onde não deve dentro da página) -> depois disso, podemos ajustar a largura dele em relação ao seu elemento contentor (ou seja, com uma medida de %), e se desejarmos, podemos ajustar seu padding para manter ele sempre centralizado dentro de seu div contentor.

Aprofundando o CSS

    Curiosidade sobre o "display: inline-block": Quando usado, ele conta até os espaços do documento html como espaçamento na página, então isso pode fazer com que itens como imagens apareçam debaixo de um elemento inline-block mesmo quando cabem do lado. Há duas maneiras de resolver isso: excluindo o espaçamento entre o elemento inline-block e o seu elemento próximo no documento html OU criando uma div entre os dois -> mudando o font-size dessa div para 0 e mudando o font-style dos elementos que tiverem texto manualmente.

Exemplo:

<div class="fonteZero">

                <div class="conteudoBeneficios">

                    <ul class="listaBeneficios">

                        <li class="itens">Atendimento ao cliente</li>

                        <li class="itens">Musicas foda</li>

                        <li class="itens">Strippers</li>

                        <li class="itens">Drogas</li>

                    </ul>

                    <img class="imagemBeneficios" src="img/beneficios.jpg" alt="imgbnf">

                </div>

            </div>

Com a estilização:

.fonteZero {

*font-size*: 0;

}

.conteudoBeneficios {

*width*: 64rem;

*margin*: 0 auto;

}

.listaBeneficios {

*width*: 40%;

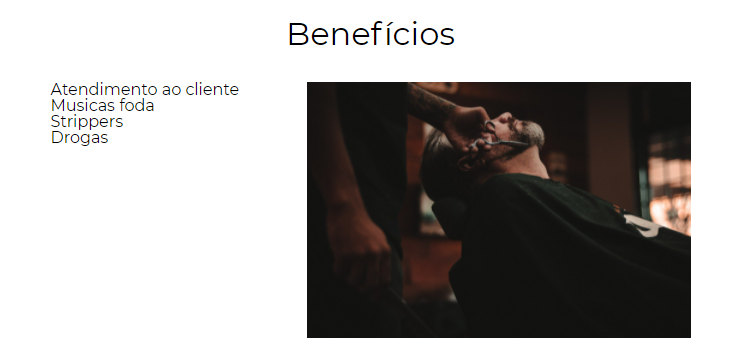
*display*: inline-block;

*font-size*: 1.6rem;

*vertical-align*: top;

}

Neste caso optei pelo segundo método para evitar mudanças no meu html, o resultado foi:



Mais sobre as Pseudo-Classes

Além das que já temos conhecimento e costume, existem também os seletores de pseudo-classes que se baseiam na ordem de descendência de um elemento em relação ao elemento pai, como a “:first-child”, “:last-child” e “:nth-child(valor numérico)”.

Este atributo basicamente permite a seleção de um elemento que esteja aninhado dentro de outro ou associado a uma classe baseado na ordem em que eles se encontram. Exemplo:

                <div class="conteudoBeneficios">

                    <ul class="listaBeneficios">

                        <li class="itens">Atendimento ao cliente</li>

                        <li class="itens">Musicas foda</li>

                        <li class="itens">Strippers</li>

                        <li class="itens">Drogas</li>

                        <li class="itens">Sodomia generalizada</li>

                        <li class="itens">Anarquia sem limites</li>

                    </ul>

                    <img class="imagemBeneficios" src="img/beneficios.jpg" alt="imgbnf">

                </div>

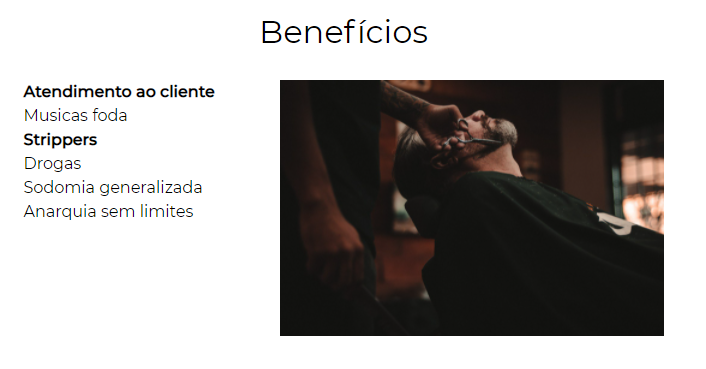
Com o estilo:

.itens:first-child, .itens:nth-child(3) {

*font-weight*: bold;

}

Gera o resultado:



Também é possível selecionar apenas os elementos pares dentro de um certo aninhamento ou pertencentes à mesma classe usando o valor 2n no “:nth-child”.

Com isso, podemos selecionar até o último elemento de um determinado aninhamento ou classe que não temos conhecimento do tamanho/largura/duração usando o “:last-child”.

Seletores Avançados

Além dos seletores de classe, id, pseudo-classes e pseudo-elementos, temos também os seletores avançados. Esses seletores permitem seleções mais específicas, então são menos usados que o resto, mas ainda assim **extremamente úteis**.

Os seletores avançados permitem a seleção a partir de um elemento de referência, sendo eles: “>”, “+”, “~”, “not()”.

- A função do ***>*** é relacionar um elemento diretamente ao seu pai na hierarquia familiar. Ele só é válido quando usado entre 2 ou mais elementos que possuam relação direta entra si, onde o primeiro precisa ser o elemento pai ou de maior hierarquia familiar, o seguinte precisa ser filho direto dele e assim sucessivamente. Exemplo:

    <main class="caixaPai">

        <h1 class="titulo">Título</h1>

        <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Ab iure accusantium assumenda inventore animi. Labore porro, nisi tempora odio consequatur laboriosam, quam molestias provident numquam at est. Eum, nobis libero?</p>

        <section class="caixaFilha">

            <h2 class="titulo">Título</h2>

            <img src="imagem1.jpg" alt="Johan" class="johan">

            <p class="netinhos">Lorem ipsum dolor sit amet consectetur adipisicing elit. Quae tempora saepe id neque rem veniam pariatur doloribus ullam, sit hic odio esse! Quod aliquid architecto accusamus voluptatem asperiores repellat similique. Lorem ipsum dolor sit amet consectetur adipisicing elit. Veniam quos repudiandae impedit quae vitae harum exercitationem recusandae nesciunt distinctio magnam nam explicabo, inventore assumenda necessitatibus! Nesciunt blanditiis eos reprehenderit id! Lorem ipsum dolor sit amet consectetur adipisicing elit. Natus inventore, placeat necessitatibus, quod illo voluptatibus quaerat iste vero ea incidunt nisi architecto quae! Exercitationem dicta consequatur itaque recusandae veritatis fugiat!</p>

            <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Rerum, laboriosam. Rerum iusto itaque excepturi corrupti consequatur earum quisquam delectus odio accusamus doloribus. Iusto mollitia nihil possimus qui, iste distinctio optio. Lorem ipsum dolor sit amet consectetur adipisicing elit. Autem eligendi assumenda dolor beatae libero explicabo consequatur, laborum saepe enim doloremque. Nisi recusandae aliquam quia quasi nihil voluptatum quam, optio facilis.</p>

            <p class="netinhos">Lorem ipsum dolor sit amet consectetur adipisicing elit. Adipisci vitae perspiciatis unde? Velit dolorem est sequi rem obcaecati sunt, ipsum tempore pariatur quae beatae in perspiciatis sit animi ratione necessitatibus! Lorem ipsum dolor sit amet consectetur, adipisicing elit. Accusamus voluptates quod aut accusantium distinctio dolore sed perspiciatis impedit consequatur. Aliquid provident illo tempora voluptatum impedit non ducimus corrupti excepturi saepe.</p>

        </section>

    </main>

Com base nesse código, podemos fazer uma estilização para todos os parágrafos dentro da .caixaPai com o seletor “.caixaPai p: {propriedades}”, mas às vezes queremos algo mais específico que isso. Podemos estilizar somente o parágrafo que é descendente direto da .caixaPai usando o seletor “.caixaPai > p: {propriedades}”. Seria possível, também, estilizar apenas o primeiro parágrafo depois do elemento img usando o seletor “img + p: {propriedades}”, e também podemos selecionar todos os elementos p que vêm depois de img com o seletor “img ~ p: {propriedades}”. Por fim, se por algum motivo quisermos selecionar todos os p que não têm a classe “netinhos” basta usar o seletor “.caixaPai p:not(.netinhos)”.

A Função Calc

As propriedades de tamanho (como largura e altura) atualmente aceitam cálculos, além das várias medidas que já vimos como valores anteriormente. Isso se torna possível usando a função calc(operação matemática) como valor.

Definindo a Opacidade no CSS

A opacidade é a ausência de transparência e ela varia entre 0 e 100%, que no CSS são representados como 0 e 1, sendo quantidades comoo 55% representadas como números decimais entre 0 e 1 (nesse caso, 0.55). Para definir a opacidade basta usar a declaração “opacity: valor entre 0 e 1”. Isso pode permitir também que elementos apareçam sobre outros em devidas circunstâncias.

A Propriedade Sombra

No CSS é possível inserir sombras nos elementos com a propriedade “text-shadow” ou “box-shadow”. Essas propriedades aceitam 4 valores: **deslocamento vertical** (aceita as mesmas medidades que width e height), **deslocamento horizontal** (aceita as mesmas medidades que width e height), **intensidade do blur** (aceita as mesmas medidades que width e height), **cor**, e o valor opcional ***inset***, que faz a sombra ficar para o lado de dentro do elemento. É possível inserir mais de uma sombra num elemento separando os valores de cada sombra com vírgulas. Exemplo:

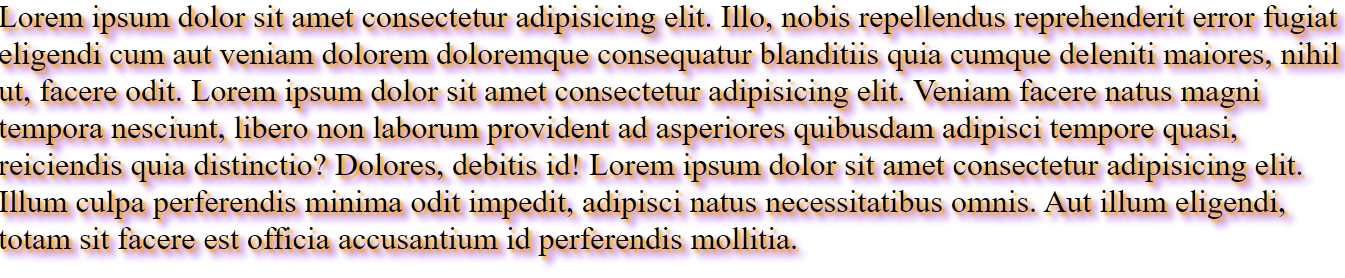
        p {

*font-size*: 2rem;

*text-shadow*: 3px 3px 3px Orange, 7px 7px 7px blueviolet;

        }

Que gera o seguinte resultado



O que é viewport?

Na computação gráfica, a viewport é a porção de área visível em um plano. Quando falamos em web responsiva, a viewport é a área visível por um visitante de uma página. O conteúdo que não pode ser visto só fica visível com um scroll, por exemplo. A viewport varia de dispositivo para dispositivo, e também varia quando o visitante que usa o desktop redimensiona a tela em que a página está sendo apresentada. No HTML 5, o viewport na tag meta deixa o desenvolvedor ter controle da área visível do navegador.

A tag <meta>

Esta tag tem como função definir a metadata sobre um documento HTML. Metadata é, no caso, a informação sobre dados incluídos no documento html, essa informação então é interpretada pelo navegador e outras ferramentas que fazem uso dela. Há outros elementos para informação de conteúdo no html, mas o meta cobre tipos de informações que os elementos como title, base, link, script ou style não podem cobrir. O elemento aceita uma grande variedade de atributos, sendo alguns deles:

## Charset

Este atributo serve para informar qual vai ser a codificação de caracteres usados na página (o padrão para o BR, com todos os acentos e símbolos necessários, é o UTf-8). Caso em nosso documento html possua algum caracter que não é comportado pela codificação especificada no meta charset, esse caracter não será interpretado da forma que esperamos.

## Refresh

Este atributo especifica a quantidade de segundos para a página ser recarregada (se for usado com o atributo “content” tendo apenas um valor numérico positivo OU a quantidade de segundos para que a página redirecione para outro endereço (se for usado com o atributo “content” tendo um valor numérico positivo seguindo do atributo url=(urlvalida.com)”.

## Viewport

Serve para dar dicas sobre o tamanho inicial do viewport. Os valores para o content do device-width podem variar entre:

### width

Que tem como função definir a largura em pixels de um viewport aceita valores numéricos positivos ou o valor literal “device-width”.

### height

Que tem como função definir a altura em pixels de um viewport. Aceita valores numéricos positivos ou o valor literal “device-height”.

### initial-scale

Define a relação entre a largura do dispositivo (largura do dispositivo no modo retrato ou altura do dispositivo no modo paisagem) e o tamanho da janela de visualização. Aceita um número positivo entre 0.0 e 10.0.

### maximum-scale

Define o zoom máximo suportado pela página. O valor usado deve ser igual ou maior ao valor usado no min-scale, ou o comportamento era indeterminado.

### minimum-scale

Define o zoom mínimo que pode ser aplicado à página. O valor usado deve ser igual ou menor usado no max-scale, ou o comportamento será indeterminado.

### user-scalable

Define se o usuário pode dar zoom na página. O valor padrão é “yes”.

Estilos Responsivos

Quando definimos um viewport para device-width, podemos ajustar estilos diferentes para tamanhos de telas diferentes. Isso é possível usando mediaqueries no CSS (que vamos ver mais sobre depois). Uma maneira de fazer isso é incluindo a regra “@media” seguida do tipo de tela que se está usando (que pode variar entre screen – para telas, print – para impresões, entre outros), seguido do seletor and() para especificar tamanho de tela para o qual estamos estilizando naquele momento. Exemplo:

@media screen and (*max-width*: 480px) {

    body {

        ...

    }

    main {

        ...

    }

}

Curiosidades Gerais

O navegador não diferencia espaços ou quebras de linhas no código html, por isso todo uso de mais de um espaço ou Enter no código sempre vai resultar em apenas um espaço na visualização da página. Para criar um espaço em branco precisamos usar o código de caracter “&nbsp;”, enquanto pra quebrar uma linha precisamos usar a Tag <br/> (ou apenas <br>).

A diferença básica entre o “display: none” e o “visibility: hidden” no CSS é que o display:none vai fazer seu elemento realmente desaparecer dali, mas o visibility:hidden apenas vai ocultar a visibilidade dele, fazendo com que os outros elementos da página permaneçam na mesma posição que estavam quando ele desaparecer.

Para melhorar a acessibilidade nos nossos forms, é ideal colocar um placeholder que identifique o input toda vez que possível, já que os leitores de tela não lêem o conteúdo escrito no elemento label.

É possível carregar arquivos de legenda nos vídeos que upamos em nosso site, basta usar o elemento <track> com os atributos src, srclang, kind e label, sendo que o texto que aparece na seleção da legenda é o valor que está em label.

