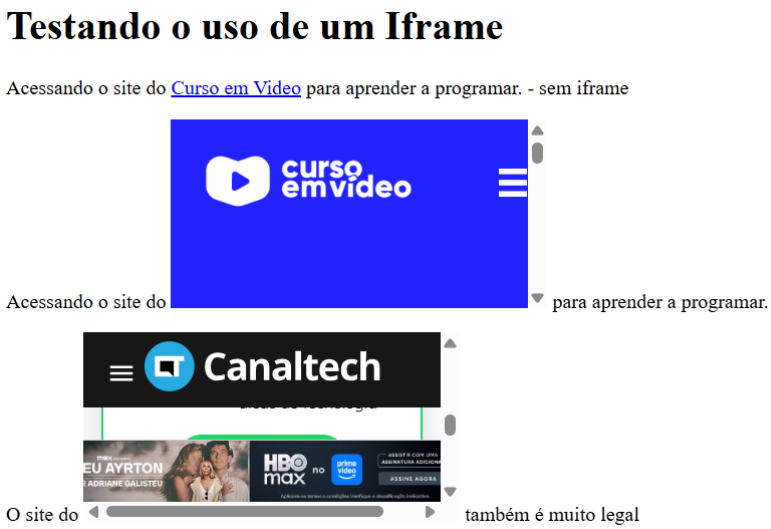
MÓDULO 4 – HTML5 E CSS3

Primeira aula – O iframe ainda pode ser usado?

O que é? Muitas pessoas acham que este i no início é frame improved (melhorado). Iframe não é um frame melhorado! Este I tem o significado de ***Inline***. Ou seja, um ***Frame Inline***. Em português: ***Quadro em linha***.

Iframe basicamente é abrir uma nova guia dentro do site. Ele tem algumas limitações, como sites que não deixam seu conteúdo ser exibido em sites de terceiros. Basta usar a tag *iframe* e colar o link desejado.



Segunda aula – Configurando Iframes

Agora será ensinado como fazer configurações básicas no iframe, como altura, borda, largura, visibilidade.... Enfim, coisas ligadas ao CSS.

Pode-se colocar uma mensagem dentro dos Iframes para que quando não forem compatíveis com determinado dispositivo, esta mensagem seja mostrada. O professor indica colocar um link no lugar da mensagem que o navegador não suporta iframe.

```
<h1>Segunda aula</h1>
<p>

    O repositório do

    <iframe src="https://github.com/gustavoguanabara"
    frameborder="0">
        [Infelizmente o seu navegador não é compatível! Sem
        problemas, clique <a href="https://github.com/
        gustavoguanabara">AQUI!</a>
    </iframe>

    também é bem legal

</p>
```

NÃO ESQUECER DE ENVELOPAR COM PARÁGRAFO!

PERSONALIZAÇÃO DE IFRAMES: Por padrão, os Iframes possuem 300x150px de tamanho. Pode ser modificado com uma tag inline, como height ou width, porém, as CSS's são soberanas. Ou seja, se houver alguma alteração na linha do iframe, que nas CSS's se confrontam, as marcações feitas dentro das CSS's irão prevalecer. E se caso não houver nenhuma destas alterações, a padrão prevalece. QUANDO REMOVEMOS O *FRAME BORDER 0* DO IFRAME, ELE DEIXA BORDA.

Como os sites selecionados são maiores do que os Iframes, eles criam a rolagem para ver o conteúdo. Para tirar a rolagem (em alguns navegadores), basta usar a tag *scrolling="no"*. No frame border, só é aceito valores entre 0 e 1. Ou seja, qualquer outro valor drasticamente diferente não fará alterações na borda do Iframe. Para modificar as bordas do Iframe, usa-se as CSS's com a tag normal *border*.

A utilidade do Iframe é colocar um conteúdo próprio dentro do site! Assunto então para as próximas aulas.

Terceira aula – Conteúdo LOCAL no Iframe

Para usar um conteúdo local para um Iframe, basta fazer uma nova página html e colocar o nome da mesma no link do Iframe.

A vantagem do Iframe é usar formatações e configurações diferentes de site dentro do que está em uso, para colocar outros conteúdos.

Quarta aula – Navegação no Iframe

Uma pequena correção referente a aula anterior: Envelopar com a tag *p* a mensagem que será exibida para navegadores que não suportam Iframe. Segunda coisa, pode-se adicionar uma pasta dentro da pasta de seu projeto para organizar os sites que serão usados para o Iframe. O que muda é que será necessário dar o caminho inteiro para se acessar a pasta. Se caso for URL com domínio próprio, basta colocar o link inteiro que relacione com o website que quer ser colocado dentro do Iframe;

Falando então sobre *navegação no Iframe*. O professor ensinou maneira de criar navegação dentro de um site. Para isso, é aberto listas com links direcionando para as páginas desejadas. Após isso, é aberto a tag de Iframe e feita a personalização do mesmo. Após finalizado as listas com os links direcionando para as páginas desejadas e o Iframe personalizado, dentro da primeira tag do Iframe DEVE ser aberto uma tag *name* e nomear ela de acordo com a sua necessidade, como um id. Os links provavelmente estão direcionando para páginas externas que abrem dentro do mesmo arquivo html, já que não foi colocado *target blank*. Entendendo o conceito do *target blank*, será aplicado um *target individual* para cada link desta lista direcionando para o *name* que foi colocado dentro do Iframe. Após isso, a abertura das páginas linkadas nas listas serão reproduzidas dentro do Iframe que foi nomeado.

```html
<ul>
    <li><a href="../ex002/pg001.html" target="frame">Primeira página</a></li>
    <li><a href="../ex002/iframe002.html" target="frame">Segunda página</a></li>
    <li><a href="../ex001/iframe001.html" target="frame">Terceira página</a></li>
</ul>
```

```html
<iframe name="frame">

    <p>Infelizmente o seu navegador não é compativel com isso!</p>

</iframe>
```

Faça uma escolha

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Deserunt debitis numquam ullam. Nobis perferendis impedit nesciunt labore illo debitis, reiciendis fugit harum ab ipsam necessitatibus repellendus suscipit perspiciatis quod recusandae? lorem

- Primeira página
- Segunda página
- Terceira página

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Excepturi a provident nesciunt quaerat beatae sit quas tempora commodi, adipisci quia soluta in asperiores quos quisquam nulla veniam veritatis iste? Fugi Lorem, ipsum dolor sit amet consectetur adipisicing elit. Ex reiciendis nulla pariatur nihil, non alias rem dolor quisquam illo? Distinctio est quia minus dolor hic architecto ducimus ipsum ipsa eius.

Faça uma escolha

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Deserunt debitis numquam ullam. Nobis perferendis impedit nesciunt labore illo debitis, reiciendis fugit harum ab ipsam necessitatibus repellendus suscipit perspiciatis quod recusandae? lorem

- Primeira página
- Segunda página
- Terceira página

Lorem ipsum dolor sit amet consectetur, adipisicing elit. Excepturi a provident nesciunt quaerat beatae sit quas tempora commodi, adipisci quia soluta in asperiores quos quisquam nulla veniam veritatis iste? Fugit Lorem, ipsum dolor sit amet consectetur adipisicing elit. Ex reiciendis nulla pariatur nihil, non alias rem dolor quisquam illo? Distinctio est quia minus dolor hic architecto ducimus ipsum ipsa eius.

Tabelas em telas pequenas

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Ipsum animi natus quaerat. Soluta voluptas dolore ab. Doloremque, iure libero? Optio debitis quod, fugit error voluptas consequuntur ea iusto voluptates deleniti. Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloribus nostrum rerum sit quia necessitatibus vel obcaecati illum. Fugiat, doloremque? Sunt fugiat libero earum necessitatibus dolorem voluptatem eveniet dicta? Unde, iure. Lorem ipsum dolor sit amet consectetur adipisicing elit. Quia facere dolorem recusandae labore deserunt aut repellat velit sint mollitia vitae? Officiis culpa quo eos rerum error pariatur quidem ipsum minima? Lorem ipsum dolor sit amet consectetur, adipisicing elit. Eaque culpa, excepturi hic maiores cupiditate incidunt soluta, pariatur tempora facilis officia ipsum similique velit et, aut molestias ipsa praesentium rem deleniti.

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Ipsum animi natus quaerat. Soluta voluptas dolore ab. Doloremque, iure libero? Optio debitis quod, fugit error voluptas consequuntur ea iusto voluptates deleniti. Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloribus nostrum rerum sit quia necessitatibus vel obcaecati illum. Fugiat, doloremque? Sunt fugiat libero earum necessitatibus dolorem voluptatem eveniet dicta? Unde, iure. Lorem ipsum dolor sit amet consectetur adipisicing elit. Quia facere dolorem recusandae labore deserunt aut repellat velit sint mollitia vitae? Officiis culpa quo eos rerum error pariatur quidem ipsum minima? Lorem ipsum dolor sit amet consectetur, adipisicing elit. Eaque culpa, excepturi hic maiores cupiditate incidunt soluta, pariatur tempora facilis officia ipsum similique velit et, aut molestias ipsa praesentium rem deleniti.

Tabelas Responsivas

Produto	Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
Notebook Profissional	120	135	142	150	160	170	165	172	180	190	200	210
Mouse Sem Fio	300	320	340	360	380	400	390	410	430	450	470	500

Quinta aula – Conteúdo no Iframe por código

Neste momento, iremos dar um complemento à aula anterior, no exercício anterior. Foi somente alterado os links dentro das listas para este exercício.

Por padrão, o Iframe quando retirado a tag *src* e *frameborder,* é aberto com um espaço em branco vazio. Para que isso não aconteça, pode-se abrir um src dentro do mesmo, e colocado uma página padrão de abertura. Ou até mesmo, colocar a primeira página das linkadas para abrir por padrão.

Entretanto, o professor ensina um novo parâmetro. No lugar de *src*, ele abre um *srcdoc* e pode formatar como um documento html. Como neste exemplo abaixo com o código e o resultado:

```
<iframe srcdoc="<h1>Escolha algum para abrir</h1>" name="frame">

    <p>Infelizmente o seu navegador não é compatível com Iframes</p>

</iframe>
```

Pode-se também adicionar mais itens, como parágrafos, subtítulos... Sempre um ao lado do outro fechando e abrindo as tags. Pode-se também, adicionar imagens, entretanto, sempre lembrar de usar SOMENTE ASPAS SIMPLES, já que o conteúdo em si está envelopado em aspas duplas, caso contrário, não irá funcionar. Obviamente, o recomendado é criar uma página separada e estilizada.

Sexta aula – Inconvenientes do Iframe

Neste momento, será feito um leve aprofundamento no conteúdo de Iframes. Será fornecido dicas, problemas de usar, e coisas semelhantes.

Primeira coisa ressaltada pelo professor: “Não confunda Iframe com frame!” – Frame é coisa do passado, ele é do html4, ainda tem gente que usa, proém o ideal seria evitar. Os Iframes sim, são úteis em alguns casos.

O professor começa a trazer alguns problemas a tona, como o robô de verificação do google não consegui entrar dentro de um iframe. Este robô, é responsável por entrar dentro dos sites, avaliar e indexar o documento contido dentro dele e determinar uma relevância para o site. A comunidade google diz que PODE ACONTECER, ou seja, NÃO É REGRA, que o robô não consiga indexar o conteúdo contido DENTRO do Iframe, e assim, tirando certa relevância do site. Indicado então é não encher de Iframes o site, para diminuir as chances de isso ocorrer.

O segundo problema é com a usablidade e acessibilidade. Acessibilidade é acesso para pessoas especiais (com algum tipo de deficiência, exemplo, pessoas cegas). As pessoas cegas utilizam um software que lê as telas. Esses softwares podem ter o mesmo problema do robô citado acima, e acaba não conseguindo acessar o iframe.

Além dos problemas de acessibilidade citados acima, também se tem os de usabilidade. Que são aqueles que o próprio navegador ou usuários acabam se confundindo. Um dos problemas é por exemplo, estar navegando dentro do iframe do curso em video e quiser retornar para a home, alguns navegadores não sabem se volta na página do conteúdo principal ou do iframe. Outra coisa, é abrir links com *target blank* dentro do iframe, que o navegador não entende que tem que abrir uma nova página na aba principal e não dentro do iframe.

Outro grande problema, se o site que estiver dentro do iframe não é resposivo, acaba aparecendo todo quebrado dentro do iframe.

Segurança: os iframes acabam abrindo uma porta dentro do seu site. Tem de ter cuidado com quem você abre a porta. Se colocar um site que foi hackeado para roubar dados, os dados do teu site também vão para o outro site. Isso é responsabilidade do criador do site.

Em alguns momentos eles são úteis, e essenciais. Mas sem abusar.

Sétima aula – Tornando Iframes mais seguros

Esta aula é referente a solução do problema de segurança dos Iframes citados na aula anterior.

Foi criado durante a aula uma página de login. Depois criado uma página com um iframe para esta página de login. Para evitar que os dados dos clientes deste site não sejam enviados para este formulário dentro do iframe, é usado uma tag simples no iframe: *sandbox=”sandbox”;* Com isto eu crio uma proteção um pouco mais seguro. Para deixar ainda mais seguro, outro parâmetro é adicionado: *referrerpolicy=”no-referrer”;* Isso diz que o conteúdo dentro do iframe não irá coletar nenhum dado do cliente deste site.

Agora, supondo que você é dono do site e quer coletar informações mesmo, dentro do *sandbox*, é possível permitir certos comando, como no vídeo: *allow-same-origin (permitir coisas da mesma origem) + allow-forms (permitir formulários) + allow-script (permitir scripts)*

```
<iframe src="index.html" frameborder="0" sandbox="allow-forms allow-same-origin
allow-scripts" referrerpolicy="no-referrer">
```

Depois foi realizado outro exercício com JavaScript, que foi possível ver como realmente funciona isso.

Oitava aula – Dicas para Iframes melhores

Foi mostrado várias maneiras de usar Iframes no site através de aplicações que usam o Iframe como base: Mapas do Google e Waze, Vídeos, Apresentações de slides (google)...

Nona aula – Como criar formulários em HTML5

Formulário é o local onde o cliente pode colocar dados. Nem sempre é digital. Antes de começar a falar sobre, será visto como funciona formulários. Formulário pode ser uma tela de login, uma aba de pesquisa para produtos dentro do mercado livre, procurar um domínio na hostnet.

Foi produzido então o formulário mais simples possível através das tags “inputs” e só temos um problema, as ligações entre “nome” e a caixa de texto não existem, e isso é ruim para os mecanismos de busca do google. Sendo assim, será ensinado isso mais para frente, juntamente para onde os dados estão indo.

Décima aula – Usar Label vai melhorar seus formulários

Na última aula foram identificados alguns problemas: as informações do formulário não iam para lugar algum, não havia ligação entre campos e nomes.

Primeira coisa a ser ensinada: tirar a sugestão dentro da caixa de texto. Para isso, ir dentro da tag form e colocar *“autocomplete=’off’.*

Segunda coisa: para onde estão indo os dados? Geralmente, esses dados são enviados para alguma linguagem de programação: Java, php…. No curso, o professor usa PHP. Dentro de form, ele abre uma tag chamada *action* e ele colocar dentro dela *cadastro.php* para mandar para uma página em linguagem de php.

Terceira coisa: ligar caixas de texto com os respectivos números. Para isso, é muito importante saber como funciona os *“nabels”* ou em português, etiquetas. As tags *name* e *id* dentro dos *inputs* possuem diferença. As tags *name* são comumente mais úteis para as linguagens html e php, por outro lado, *id* é mais útil para JavaScript e *labels*. Para então se usar as etiquetas, se envelopa o termo, neste caso deste formulário de envio de nome e sobrenome, usaremos o nome para exemplificar. Envelopa-se o termo *nome* com a tag *label* e dentro de *for* se coloca então o *id* do respectivo lugar que será conectado, neste caso, a caixa de texto referente ao *nome*. Então, o professor fez um combinado que

durante todo este módulo, será colocado antes de qualquer identificação dentro de *id* a letra *i.* Ou seja, o ID da caixa de texto para colocar nome, estava escrito somente *“nome”* e agora será identificada por *“inome”,* e isso para todos os termos que necessitam de um ID dentro dos formulários.

NÃO SE FAZ FORMULÁRIO SEM RELACIONAR “LABELS” COM “INPUTS”!! Isso é para os mecanismos de busca do google funcionarem.

Ao enviar os dados, pode-se perceber que não abrirá outra página, e os dados referentes ao formulário se encontram dentro da URL do site. Isso não é bom. Iremos aprender como resolver isso na próxima aula, onde o tema é métodos de envios de dados através de formulários.

Décima primeira aula – Métodos GET e POST para formulários

Existe um padrão de formulários que faz com que os dados apareçam na URL do site, ou seja, é um método de envio pela URL. No HTML se têm dois métodos simples para o envio de formulários. O primeiro método, é este padrão que está sendo utilizado: *“get”*. Quando não se define nenhum método de envio, se aplica o normal, o padrão: o famoso *“get”.*

Para alterar as opções de envio de formulários, necessita-se que abra dentro da tag *form* a tag *method,* e lá, se encontra as opções de envio: GET e POST. O padrão é GET. Se tirar o GET e colocar POST, os dados enviados para o formulário não aparecerão na HTML. Porém, não é porque não aparece na URL os dados que esse método é o melhor e mais seguro.

Existe uma maneira de ver ainda esses dados, eles só não “estão jogados na cara do usuário”. Para ver, basta abrir o inspecionar, e lá dentro e clicar em *network.* Lá dentro, digitar dentro do formulário (no site) os dados e enviar. Após enviar, aparecera o arquivo php e ao clicar nele e em *headers* se pode ver os dados enviados. Para proteger os dados, necessita-se criptografar com HTTPS os dados, nesse caso, estão sendo usado o HTTP. O POST só está enviando os dados de outra maneira, mas ainda se pode ver os dados. Ou seja, ele NÃO PROTEGE OS DADOS.

E então fica uma dúvida gigante: em que momento se usa GET e em que momento se usa POST? GET: dados não são sensíveis. Nome, peso.... Não está pedindo senha, endereço, CEP, cartão...

Outra coisa importante: quando se envia dados por GET, só pode enviar até 3.000 bites. Se ultrapassar isso, ele não serve. 3.000 bites seriam 3k de letras. Outro case que não se usa o GET: campos sensíveis (senha, cartão, endereço) e também envio de fotos e qualquer tipo de arquivo. Caso contrário usa o POST.

Para POST: dados sensíveis, entretanto, o ideal é aprender HTTPS para proteger dados. Se os dados tiverem mais de 3k bites. E por fim, se quiser fazer envio de arquivos.

Décima segunda aula – Criando caixas de texto e senha

Agora, serão apresentados vários controles e muitos atributos relacionados a formulários em HTML. Agora, será colocado controles especiais para formulários. Nós não vamos aprender cem por cento dos controles, pois alguns não são usados, outros não são compatíveis com navegadores. Serão aprendidos os essenciais.

Detectado um problema: caixas obrigatórias. No último formulário, se caso o usuário não digitar nada e clicar em enviar, ele consegue enviar. Mas antes disso, para não ficar recebendo a mensagem de erro toda a vez: na pasta criar um arquivo *cadastro.php* e colocar: “Os dados foram enviados” e abaixo: “aprenda PHP para saber o que fazer com eles. ” Porém, não é para enviar os dados vazios. Será criado um arquivo novo, um novo formulário. Nesse formulário criar o código base, e fazer um formulário. Dentro dele, um parágrafo ou div, e colocar três caixas: texto para nome, senha, enviar e limpar. Nome e senha com etiqueta. Para ser obrigatório: *required* dentro de imput. Colocar o POST porque pede senha. Para determinar quantidade de caracteres se usa: *minlength e maxlength*. Na senha, geralmente se diz máximo e mínimo. Para mudar o tamanho da caixa, se coloca dentro de *input* a tag *size* e coloca um valor, ex. 10. Isso diz que por exemplo, a caixa aguenta até 10 caracteres. Se

for determinado quantidade de caracteres com *min/maxlength* significa que: a caixa aguenta até x quantidades de caracteres (min/max) e só irá mostrar 10 desses caracteres.

Para dar aquelas dicas do que escrever dentro, ex. Nome de Usuário, ou da senha, abrir dentro do input um tag *placeholder* e colocar a dica desejada.

Os navegadores salvam os nossos dados referentes a determinadas caixas, como a de usuário. Porém, precisamos ligar o autocomplete no *form* e dentro de cada *input* sinalizar o que essa caixa quer. Então dentro dela, se abre um *autocomplete* após ligar o *autocomplete on* no form, e colocar na de usuário, por exemplo: *username*. Porém, tem várias opções. Para senha, se tem duas opções: *newpassword* (cadastro novo, se cadastre) ou se quiser a senha de login, atual, usa a *corrente-password*.

Na próxima aula, aprenderemos novas caixas mais interessantes.

Décima terceira aula – Elementos number, month, date e time em formulários HTML

Min e max são para números, quando tem o length é para caracteres. Se quiser que já apareça alguma coisa: *value* e o valor. Isso tira o placeholder.

Foi apresentado então diversas caixas com os conceitos acima. Ex009.

Décima quarta aula – Compatibilidade com navegadores

Professor fez o levantamento de uma pergunta referente as aulas: quando os alunos retornam para suas casas (nos cursos presenciais), e tentam fazer (treinar) o que fizeram em aula, porque a tela fica diferente?

Professor abriu vários navegadores e foi mostrando o último formulário feito (mudança de ícones percebidas, alguns comandos não funcionam) e mostrou os diferentes comportamentos relacionados. O MacBook foi o que reagiu pior com data e hora. Várias coisas erradas. Ou seja, se chega na conclusão que nem sempre irá funcionar em todos os navegadores.

Não se tem muito o que fazer por enquanto, já que não temos uma base de JavaScript, que vai fazer com que o desenvolvedor não se prenda e dependa a navegadores. Ou seja, o desenvolvedor desenvolver os próprios controles.

O objetivo desta aula, é mostrar as diferentes compatibilidades dos navegadores. Os navegadores têm o padrão de ir se atualizando. Sempre verificar em todos os navegadores para ver as diferentes perspectivas acerca do site produzido.

Décima quinta aula – Formulários com telefone e e-mail

Criação de um novo formulário. Para colocar o e-mail: *input: email.* Para colocar telefone: *input: tel.* Estudar mais sobre: RegEx! Dentro de html, se pode colocar um RegEx dentro de um parâmetro chamado *pattern*, sempre começando a expressão com uma exclamação e termina com um cifrão.

Para finalizar, o professor vai adicionar um novo parâmetro, um agrupamento de campos. Basta selecionar todos os elementos do form, e envelopar em uma tag chamada *fieldset* e colocou uma *legend* colocando por exemplo: *Dados Pessoais.*

Décima sexta aula – Checkbox e Radio button em HTML

Nesta aula, professor apresenta dois comandos novos. Criar novo exercício para começar.

Primeiro comando ensinado foi o de *Checkbox*, que é determinado pela tag *input: Checkbox* e uma coisa muda. Antes era colocado o *label* acima do parâmetro *input,* agora, será colocado primeiro o input, ao lado do input o termo associado envelopado dentro de uma *label*

```
<fieldset>
    <legend>Esportes Favoritos</legend>
    <input type="checkbox" name="esfut" id="ifut"> <label for="ifut">Futebol</label> <br>
    <input type="checkbox" name="esbasq" id="ibasq"> <label for="ibasq">Basquete</label>
    <br>
    <input type="checkbox" name="esping" id="iping"> <label for="iping">Ping-Pong</label>
    <br>
    <input type="checkbox" name="esvolei" id="ivolei"> <label for="ivolei">Vôlei</label>
    <br>
</fieldset>
```

O segundo parâmetro ensinado é o *radio*, onde é delimitado por *input: radio* e é aquela caixa, por exemplo, de sexo, onde você só pode escolher um. A diferença dele e do *Checkbox* é esta. Entretanto, quando abrir esta tag, deve-se colocar o mesmo *NAME*, e os id pode ligar normalmente com as palavras. Por quê? Se não fizer isso, é possível marcar as duas caixas de texto e não conseguir mais desmarcar. E outra coisa, neste caso, necessita-se somente de uma opção, não as duas. Se caso quiser deixar um já marcado, basta ir no qual desejar deixar marcado e escreve *checked*. Assim ao abrir, os marcados já ficam marcado. TODO O BOTÃO DE *RADIO* DEVE TER O MESMO *NAME*, ENTRETANTO A ID DEVE SER DIFERENTE PARA RELACIONAR COM O LABEL. PARA OS DADOS DE *RADIO* TEREM VALOR MAIS ENTENDÍVEL NO DADO, COLOCAR VALUE M OU F PARA OS SEXOS.

Décima sétima aula – Elementos color, range e file em HTML

Novo formulário. Primeiro parâmetro, é colocar cor, informar cor de alguma coisa. Basta usar o *input: color*. Dá para escolher várias cores, podendo definir parâmetros específicos, como por exemplo, já ter uma selecionada: *value*.

O outro que foi ensinado é o range: *input: range* e pode ser usado para por exemplo, nível de satisfação do usuário. Pode-se, também, definir parâmetros específicos: *min* e *max*. E onde começar também: *value.*

O último controle, um lugar para subir arquivos, como por exemplo, para definir foto de perfil. Basta usar: *input: file.* Assim, o usuário pode enviar arquivos. Porém, quando usar este tipo de controle, NÃO SE PODE USAR O TIPO GET PARA ENVIO, E SIM, POST!!!

Décima oitava aula – Select, datalist e textarea em HTML

Agora será ensinado alguns elementos que são criados sem depender da tag *input.* Possuem tags específicas.

Criar novo arquivo. Para o primeiro exemplo, será sugerido ao usuário selecionar o estado que ele habita. Para isso, seria interessante aquela caixa

que se clica e abre uma lista. O nome dela é *select*. Ela pode ser definida por: *Select* com o nome do que será selecionado, um id para relacionar com a *label*, e ela termina com *opt* para as opções. Se for interessante, pode-se agrupar as opções em grupos separados, como a região de cada estado:

```
<form action="cadastro.php">
    <p>
        <label for="iest">Estado:</label>
        <select name="Estado" id="iest">
            <optgroup value="Região Sudeste">
                <option value="SP">São Paulo</option>
                <option value="RJ">Rio de Janeiro</option>
                <option value="MG">Minas Gerais</option>
            </optgroup>

            <optgroup value="Região Nordeste">
                <option value="RN">Rio Grande do Norte</option>
                <option value="PE">Pernambuco</option>
                <option value="MA">Maranhão</option>
            </optgroup>
        </select>
    </p>
```

Se quiser já deixar um selecionado, basta escrever *selected* ao lado do elemento. É muito comum as pessoas criarem uma opção vazia de *opt* e colocar algo como *escolha* e colocar *selected* aqui, para a pessoa saber que tem que escolher uma opção.

Agora, supondo que não tenha determinada informação dentro deste *Select*. Neste caso, usamos outro parâmetro: *datalist*. Vamos supor que o objetivo e pedir a profissão do cliente: usa-se o *input: text* com uma *label* relacionando os dois, e abaixo do text, para se abrir uma caixa de opções, usa-se *datalist* com *opt*. Após finalizar as opções, se usa um id dentro da *datalist* e abre um *list=”id”* no *input: text*. Se quiser ser retirado as abreviações das profissões, somente retirar os *value*.

Para finalizar a aula: comentário, observação: *textarea*, depois basta definir a quantidade de caracteres para quebrar linha (col) e quantas linhas irão aparecer (row). Não esquecer da *label* antes.

Décima nona aula – Elemento output em formulários HTML

Nesta aula, será usado um pouco de JavaScript para usar o *output* em HTML.

Output só serve para mostrar coisas na tela, e não enviar dados.

Foi realizada algumas programações em JavaScript para usar output.

Vigésima aula – O que são Media Queries em CSS

O objetivo desta aula é explicar para que serve uma Media Query. De maneira bem objetiva: O site que um desenvolvedor produz pode ser exibido em diferentes tamanhos de tela, como celulares, notebooks e monitores. As Media Queries permitem adaptar o layout e os estilos desse mesmo site para diferentes telas e também para impressão, tornando-o responsivo.

Após a explicação, foi realizado os downloads dos conteúdos relacionados ao módulo.

Vigésima primeira aula – Criando um site com versão para impressora

Foi criado um site simples para um noticiário linkando dentro de *head* dois tipos de CSS, um para impressoras, o qual foi apelidado de *print* e o outro para as telas em gerais, apelidado de *screen*. Os dois possuem configurações diferentes, já que possuem objetivos diferentes. O que foi aprendido aqui, é chamado de *Media Types*, e na próxima aula, será apresentado as *Medias Features*

Vigésima segunda aula – Múltiplas Media Features com CSS

Iremos falar sobre as Media Features, que auxiliam as Media Queries a ficarem completas. Para entender isso em uma fórmula: **Media Query = Media**

Type + Media Feature, sendo Media Type referente ao tipo de Media, e Media Feature, característica de mídia.

Foi criado um novo exercício. Nele foi colocado um ***h1*** escrito “Mude a orientação do dispositivo”. Foram também usadas somente duas imagens da pasta baixada no início do capítulo: Curso em Vídeo Paisagem e Retrato.

No exercício anterior, o Guanabara comentou que muitas vezes, teríamos a mesma configuração para as duas CSS, que ele iria ensinar como fazer isso de maneira mais prática ao invés de colocar duas vezes no código. Para evitar isso, abaixo de header, o primeiro link para CSS é para a *media: all*, ou seja, para todas as mídias. Logo após, ele cria dois outros links para CSS: um para o retrato com o *media: screen* e outro para a paisagem, com o mesmo media, já que os dois são referentes à tela dos dispositivos. Logo após, ao lado do tipo de mídia que está sendo usado, é adicionado um *and* e fornecido a característica (Media Feature) referente aquele CSS. Ficando, então, para o modo retrato igual a = ***media: “screen and (orientation: portrait) ”*** (Media Types são fornecidas dentro de aspas, enquanto as Media Features são fornecidas dentro de parênteses.). E para a Paisagem = ***media: “screen and (orientation: landscape) ”.*** Isso diz que (usando o link para a CSS referente ao modo retrato): **O arquivo *retrato.css* vai dar as configurações para todas as telas que estão em modo retrato. O mesmo para a paisagem.**

Vigésima terceira aula – Seguindo a orientação do dispositivo

Nesta aula, foi realizada as operações em cada CSS da aula anterior. No CSS para o *all*, foi realizada modificações visuais que seguem para todas as páginas, independente de tipo de mídia e de característica. Logo após, foram feitas modificações especificas para cada tipo de orientação de tela. Ou seja, para cada tipo de tela, elas têm uma base igual, que é definida pelo *all*, e quando mudar a orientação ou se mantem em um padrão, algumas pequenas coisas se alteram. Essas pequenas mudanças, são feitas em CSS’s diferentes.

Vigésima quarta aula – Reunindo tudo em um único CSS

Quando um projeto é pequeno, não faz sentido criar tantas páginas. Para projetos grandes, aí sim se faz necessário. Para isso, criamos um novo exercício novo copiando todo o html antigo.

Para juntar tudo em só basta ou abrir um *style* dentro do próprio HTML ou até mesmo linkar para uma única página de CSS. Dentro dele, você faz as configurações gerais dentro de uma tag: ***@media all*** e para as outras orientações, você também faz as alterações dentro de outras mídias: ***@media screen and ...*** MAS VALE RESSALTAR: ISSO DEIXA DESORGANIZADO O CÓDIGO, É MELHOR PÁGINAS INDIVIDUAIS PARA EVENTUAIS ORGANIZAÇÕES E CORREÇÕES DE CÓDIGOS.

Vigésima quinta aula – Mobile First

Esta é uma abordagem que surgiu em 2009. Muita gente desenvolvia site do jeito que estávamos desenvolvendo. Desenvolve o site do jeito que queria e depois vai tirando coisas para deixar ele móvel, ou em outros termos: responsivo.

A proposta dele foi: ao invés de construir todo o site, e começar a tirar coisas para deixar ele responsivo, o que faz perder experiência do usuário por ficar retirando essas coisas, porque não primeiro construir a versão móvel e depois reposiciona as coisas sem ficar removendo tantas *Features*. Depois que a versão móvel fica ok, daí remodela.

O criador dele, trabalha no Google, ele disse que se fazer isso, além de melhorar a experiência do usuário, você também é mais facilmente reconhecido pelo algoritmo do Google. Isso também, aumenta a credibilidade do seu site.

Logo após a explicação, baixamos e organizamos as imagens que serão usadas nos próximos projetos.

Vigésima sexta aula – Iniciando um site mobile First

Fizemos um projeto de site criando dois CSS, um que é o *all* (mobile First) e outro sem mídia nenhuma, chamado de *style.css* mesmo, que se refere a todos os outros que iremos organizar com as tags ***@media***.

Vigésima sétima aula – Device BreakPoints

Quando criamos uma Media Query, precisamos definir tamanhos para telas: tamanho da tela do tablet, do celular, do computador, o limite da tela de um monitor. Porém, isso fica complicado. Ele vai variando conforme o passar de anos. Para isso, se pesquisa sobre o assunto.

Retornando então ao exercício anterior. Temos somente a versão do celular.

Para criar diferentes tipos de projeções do mesmo site, se utiliza a tag ***@media*** no CSS, e com ela, se define a quantidade máxima e mínima de pixels de acordo com as regras pesquisadas. Chegamos a seguinte conclusão:

```
/*
Pesquisa por: Typical Device BreakPoints
-----------------------------------------
-> Pequenhas telas: até 600px
-> Celular: de 600px até 768px
-> Tablet: de 768px até 992px
-> Desktop: de 992px até 1200px
-> TV: 1201px acima
```

Então, com base nesses dados, vou colocar abaixo como título de exemplo, como será feita a reprodução dentro do tablet. Dentro de uma tag media, definimos qual tipo de mudança haverá (na screen do usuário nesse

caso) e depois, qual o tamanho de acordo com a tela desejada, como nesse exemplo, o tablet, e define o que muda dentro dessa tela.

```
@media screen and (min-width: 768px) and (max-width: 992px) /* tablet */ {
    body {
        background-image: url(../imagens/back-tablet.jpg);
    }

    img#phone {display: none;}
    img#tablet {display: block;}
    img#print {display: none;}
    img#pc {display: none;}
    img#tv {display: none;}
}
```

Dentro da própria tag style, se modifica os acontecimentos como se fosse cada tag media uma página de estilo. Muitas vezes, vale mais a pena criar páginas diferentes para cada tipo de tela. Nesse caso, como as mudanças eram pequenas, pode-se definir através de um único CSS todas as mudanças.

Vigésima oitava aula – Menu Responsivo

Agora será ensinado um conceito muito importante sobre responsividade. Um menu responsivo otimiza espaço para telas e ajuda a melhorar a responsividade em dispositivos.

Foi criado um código base de organização do site, com um título de nível um, dois e parágrafos com lorem. Após isso, foi criado um link para CSS e trocado a font-family.

Vigésima nona aula – Configurando o layout do projeto

Foi feito ajustes dentro do CSS, como mudança do background, animações simples, e coisas assim. Nada de complexidade.

Trigésima aula – Criando um menu hambúrguer

Para fazer o ícone do menu, usamos um site do google, pesquisando por *google icons* e achando o ícone desejado. Após gravar o nome desse mesmo ícone, temos de ir até *developer guide*, e achando a maneira de se aplicar ao seu projeto. No nosso caso, desenvolvimento web, podemos fazer um link direto para os servidores do google ao nosso código com uma importação. Após obter o código, basta importar ele junto com as CSS. Essa é uma linha que vai permitir-nos usar um ícone.

Logo após, para utilizar os ícones do google, basta, na mesma página que estava, verificar como usar os ícones. Ele dá um código pronto para usar o ícone. Pode até mesmo clicar no ícone, fazer o link de exportação somente dele (se caso tiver só esse), e depois clicar para importar ele ao código. De qualquer jeito, volte nele e clique nele para copiar o seu código. Para facilitar a configuração do mesmo, basta colocar um ID nele, já que ele já possui uma *class* de fábrica.

Para o menu funcionar, foi usado tecnologias de JavaScript.

```
<span id="burguer" class="material-symbols-outlined"
onclick="clickMenu()">menu</span>
```

```
<script>

    function clickMenu() {
        if (itens.style.display == 'block') {
            itens.style.display = 'none'
        } else {
            itens.style.display = 'block'
        }

    }

</script>
```

Trigésima primeira aula – Media Queries para outros dispositivos

Nesta aula, foi somente editada outras formas de exibição. E também, alterações em JavaScript.

Trigésima segunda aula – Tela de login responsiva só com HTML e CSS

O projeto agora é construir uma tela de login responsiva somente com HTML e CSS. Criamos a base do projeto, criando as divisões e dando pequenas identificações. Foi colocada os códigos da cores também.

Trigésima terceira aula – Versão Mobile-First da tela de login

Foi feita toda a preparação do mobile First, colocando no centro, editando algumas cores e colocando as imagens. Foi colocado largura, altura, definição de bordas. Enfim, a parte mobile está pronta.

Trigésima quarta aula – Outras Media Queries para o projeto

Aqui foi criada a nova página de CSS que organiza as Media Queries para os outros dispositivos. Aqui também foi organizada algumas animações a respeito das mudanças de telas.

Trigésima quinta aula – Criando o formulário de login

Nesta aula foi adicionado os formulários dentro da parte de login, junto com as suas etiquetas, e também, importado alguns ícones para complementar a composição visual do mesmo.

Trigésima sexta aula – Aplicando estilo ao formulário de login

Nesta aula foi colocado alguns efeitos dentro dos *inputs* que estão dentro de uma *div* e aplicando cores, modificando tamanhos, estilos, cores. Entretanto, ainda falta formatações.

Trigésima sétima aula – Fim do projeto Login

E aqui então foi colocada cada peça no seu devido lugar. Tudo da maneira mais inteligente possível.