**DETALHES IMPORTANTES AO FAZER FORMULÁRIOS:**

**SEMPRE VALIDE OS CAMPOS**

* Campos obrigatórios devem ter o “required”;

**USE OS TIPOS CERTOS**

* Não esqueça de ver se os campos realmente estão atendendo ao tipo certo por através do atributo “type”;
* Por exemplo não permita que o um campo de email seja “type = text” ele têm que ser “type = email”;

**USE O TITLE PARA DAR DICAS**

* O atributo title dá uma dica aos usuários sobre oque eles devem escrever dentro de um determinado campo;
* Ele funciona de 2 formas, em campos sem “required” se o usuário colocar o mouse encima do campo aparece um tooltip com a dica sobre o campo. Em campos com “required” ele vai dar a dica quando o usuário não preencher o campo;

**SOBRE DATAS E FAIXA ETÁRIA**

* Em alguns formulários é necessário cumprir uma determinada faixa etária para que a pessoa possa preencher o formulário;
* Use essa fórmula para faixa etária:

const dataNascimento = document.querySelector(#id\_do\_input)

dataNascimento.addEventListener(‘blur’, (e) => {

validaDataNascimento(e.target)

}

function validaDataNascimento(input){

const dataRecebida = new Date(input.value)

let mensagem = ‘’

if(!maiorQue18(dataRecebida)){

mensagem = ‘Você deve ser maior que 18 anos para se cadastrar!’

}

input.setCustomValidity(mensagem) //Exibe uma mensagem de erro

//customizada à nossa escolha

} //se um valor vazio for passado

//o js entende que a validação

//deu certo

function maiorQue18(data) {

const dataAtual = new Date()

const dataMais18 = new Date(data.getUTCFullYear() + 18, data.getUTCMonth(), data.getUTCDate())

return dataMais18 <= dataAtual

}

Note na fórmula acima que nós usamos um evento blur sempre que o usuário termina de digitar a sua data de nascimento a função validaDataNascimento pega o valor do input da data chama a função maiorQue18 para comparar se a data de nascimento + 18 anos á frente é menor ou igual a data de hoje, se a quantidade de anos for superior a data de hoje significa que o usuário ainda não completou 18 anos, e uma mensagem será mostrada para ele.

**SOBRE SENHAS**

* Delimite um número mínimo para os caracteres de senha por através do “minlength = “6”” por exemplo;
* Delimite um número máximo para os caracteres de senha por através do “maxlength = “12”” por exemplo;
* Utilize o atributo “pattern” para definir um método de escrita que deve ser empregado, geralmente usamos uma regex para isso, dessa forma:

**“pattern = “^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*[0-9])(?!.\*[ ^!%\*¨/\()=§]).{6,12}””**

Essa é uma regex determina que a senha tenha pelo menos 1 letra maiúscula, 1 numeral e pode conter qualquer caractere especial que não seja esses: espaço^!%\*¨/\()=§;

**USE LABELS PARA FACILITAR A VIDA DOS USUÁRIOS**

* Sempre amarre o “for” do label aos ids que correspondem ao input, isso faz com que quando o cliente clicar no label ele acione o campo o input;

**DATA-ATTRIBUTES EM VEZ DE IDS**

* Em vez de ficar criando uma série de ids para cada input, crie um único data-attribute e crie variáveis que poderão ser referenciadas no javascript. Dessa forma:

data-um\_nome\_qualquer\_a\_sua\_escolha = “nome\_da\_variavel\_que\_referencia\_o\_input”

Explicando o código: data-attributes são atributos que podemos criar para gerar uma variável que captura os valores dos inputs e poderá ser referenciada por através de um nome à nossa escolha, elas devem obedecer a regra de ter um prefixo “data-” seguido por um nome à nossa escolha, e devem receber como valor um nome que identifique aquele input.

data-identificador = “nome\_que\_identifica\_o\_input”

Os data-attributes são muito úteis para identificarmos vários inputs pelo atributo invés de id, e como cada um deles recebe um nome que o identifica, podemos criar um array em JS para adicionar um comportamento específico para cada um, como por exemplo:

//Uma constante que pega todos inputs e os coloca num array...

const inputs = document.querySelectorAll(‘input’)

//Adiciona um evento blur para validação de cada input capturado...

inputs.forEach(input => {

input.addEventListener(‘blur’, (event) => {

validaInput(event.target) //cada evento chama a função

}) // validaInput

})

//A função validaInput verifica se o input passado possuí um valor de data-attribute, se tiver, ele pegará uma função específica para aquele valor data-attribute...

function validaInput(input){

const tipoDeInput = input.dataset.tipo\_de\_data-attribute //dataset

//chamamos um objeto que pode conter uma determinada chave

//contendo uma função de validação como seu valor

if(validadores[valor\_do\_data\_attribute]){

validadores[valor\_do\_data\_attribute](input)

//aqui estamos chamando a função para aquele determinado

//data-attribute e passando o evento como parâmetro

}

}

//Objeto que contém uma função de validação diferente para cada valor de data-attribute que existir...

const validadores = {

valor\_do\_data\_attribute1 = input => funcaoValidadora(input),

valor\_do\_data\_attribute2 = input => funcaoValidadora(input),

valor\_do\_data\_attribute3 = input => funcaoValidadora(input)

}

**SERVIDORES PARA TESTES:**

* Caso deseje usar servidores para testar o seu formulário você pode usar:
* Apache
* Live-Server do VSCode
* Browser-Sync
* Lembrando que o Browser-Sync é uma biblioteca Node que pode ser instalada direto no Node usando o comando:

npm i –g browser-sync

Caso a instalação não dê certo, utilize o sudo, assim:

sudo npm i –g browser-sync

Depois de aberto, digite o comando:

browser-sync start –s –f . –directory

Esse comando faz com que o servidor seja startado, será apresentado no terminal qual a porta que o localhost vai utilizar para o servidor no browser.