**Começando com fetch no Javascript**





**elias-ribeiro**

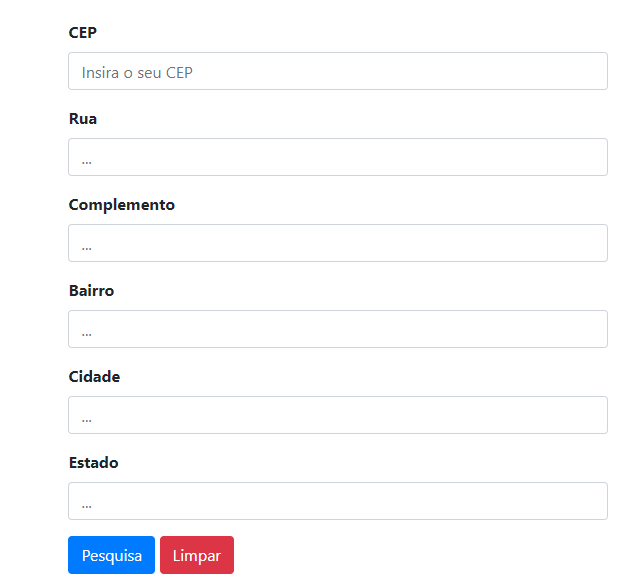
27/12/2018

**COMPARTILHE**

**Introdução**



Saiba como eu utilizei o fetch para consumir dados de outros sistemas via Rest e preencher automaticamente a seguinte lista:



Atualmente a pessoa que fez a compra precisa preencher todos os campos de maneira correta para que haja **validação** e a compra seja finalizada com sucesso.

Porém, pode ser que alguns dos campos seja digitado de maneira incorreta resultando em um endereço errado. Sendo assim teremos problemas na validação, finalização da compra e o descontentamento do cliente em relação a loja pois não conseguiu efetuar a finalização da compra.

**Automatizando**

Para automatizar o preenchimento iremos usar o [**Javascript**](https://www.alura.com.br/artigos/javascript), vamos primeiro ver como o **HTML** está definindo e pegar o atributo do button pelo Javascript.

<div class="container col-md-5">

<form>

<div class="form-group font-weight-bold">

<label>CEP</label>

<input type="text" id="cep" pattern= "\d{5}-?\d{3}" class="form-control" required placeholder="Insira o seu CEP">

</div>

<div class="form-group font-weight-bold">

<label>Rua</label>

<input type="text" class="form-control" id="rua" placeholder="...">

</div>

<div class="form-group font-weight-bold">

<label>Complemento</label>

<input type="text" class="form-control" id="complemento" placeholder="...">

</div>

<div class="form-group font-weight-bold">

<label>Bairro</label>

<input type="text" class="form-control" id="bairro" placeholder="...">

</div>

<div class="form-group font-weight-bold">

<label>Cidade</label>

<input type="text" class="form-control" id="cidade" placeholder="...">

</div>

<div class="form-group font-weight-bold">

<label>Estado</label>

<input type="text" class="form-control" id="estado" placeholder="...">

</div>

<button type="submit" id="btnPesquisar" class="btn btn-primary">Pesquisa</button>

<button type="button" id="btnLimpar" class="btn btn-danger">Limpar</button>

</form>

</div>

No Javascript vai ficar assim:

const btnPesquisarCEP = document.querySelector("#btnPesquisar");

Com isto feito, adicionamos um evento de click do botão usando **[arrow function](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow_functions" \t "_blank)**:

btnPesquisarCEP.addEventListener("click", event =>{}

Vamos implementar algumas coisas dentro dessa função, primeiro **event.preventDefault**, responsável por cancelar o comportamento natural do botão que seria uma submissão para o servidor.

btnPesquisarCEP.addEventListener("click", event =>{

event.PreventDefault();

}

Com o comportamento natural do botão cancelado, iremos implementar a busca do CEP e pegar o valor que está contido no campo do CEP.

const inputDoCep = document.querySelector("#cep");

const valorDoCep = inputDoCep.value;

Após ter feito isto, vamos começar a trabalhar com a **API** do [Viacep](https://viacep.com.br/" \t "_blank) e utilizar o **fetch** pela URL para fazer uma [**requisição**](https://www.alura.com.br/artigos/metodos_de_requisicao_do_http). Se ela for achada, vamos fazer uma função retornando a resposta em **json**.

const inputDoCep = document.querySelector("#cep");

const valorDoCep = inputDoCep.value;

const url = `https://viacep.com.br/ws/${valorDoCep}/json/`;

fetch(url)

O fetch devolve uma promessa de que algo será retornado, essa promessa é chamada de **Promisse**. Essa promessa pode tanto ser boa, ter retornado os dados, quanto ter falhado por algum motivo - como no caso da **conexão com o servidor** cair.

Para ambas as situações precisamos fazer um tratamento. No caso, só vou tratar no caso de sucesso. Por isso, se for um sucesso, então (**then**) vamos pegar a resposta com as informações do CEP:

const inputDoCep = document.querySelector("#cep");

const valorDoCep = inputDoCep.value;

const url = `https://viacep.com.br/ws/${valorDoCep}/json/`;

fetch(url).then(response =>{

return response.json();

}).then(data =>

{

})

Com o retorno da data (dados) vamos atribuir ele para os campos e fazer uma função pegar o id do campos e atribuir os valores para eles:

function atribuirCampos(data)

{

const rua = document.querySelector("#rua");

const complemento = document.querySelector("#complemento");

const bairro = document.querySelector("#bairro");

const cidade = document.querySelector("#cidade");

const estado = document.querySelector("#estado");

rua.value = data.logradouro;

complemento.value = data.complemento;

bairro.value = data.bairro;

cidade.value = data.localidade;

estado.value = data.uf;

}

E dentro do evento de click do **btnPesquisarCEP** iremos atribuir a função:

const inputDoCep = document.querySelector("#cep");

const valorDoCep = inputDoCep.value;

const url = `https://viacep.com.br/ws/${valorDoCep}/json/`;

fetch(url).then(response =>{

return response.json();

})

.then(data =>{

atribuirCampos(data);

})

Mas como sabemos, o CEP pode ser digitado de maneira errada, então vamos fazer uma exceção que mostre para a pessoa que foi um CEP incorreto ou inexistente.

Para isso criamos um alert dentro do then para mostrar que o CEP está inválido.

.then(data =>

{

if(data.erro)

{

alert("O CEP DIGITADO ESTÁ INVÁLIDO");

return ;

}

atribuirCampos(data);

})

**Conclusão**

Neste post, aprendemos como fazer uma requisição web com Javascript vendo como são úteis e como elas facilita a vida de quem desenvolve. E normalmente no dia a dia, os sistemas se comunicarem com outros por requições.

Se gostou do conteúdo, que tal dar uma olhada na nossa [**formação front-end na Alura**](https://www.alura.com.br/formacao-front-end) ou em nossos livros na [**Casa do Código**](https://www.casadocodigo.com.br/collections/front-end-javascript).