[ARTIGOS](https://www.alura.com.br/artigos) > [**FRONT-END**](https://www.alura.com.br/artigos/front-end)



**Preenchendo formulário HTML automaticamente com AJAX**

matheus.castiglioni

16/01/2017

**COMPARTILHE**

Durante o desenvolvimento de um projeto web, nosso cliente pediu que no cadastro de endereço criássemos um recurso para busca de CEP automática facilitando o processo para o usuário, mas como podermos fazer isso?

Inicialmente, poderíamos adicionar algum tipo de botão ou listener que ficasse responsável em fazer uma requisição HTTP enviando o CEP, e então, recarregasse o formulário automaticamente.

Porém, quando estamos em formulários de cadastro de endereço, como por exemplo em sites de e-commerce, é muito comum adicionarmos apenas o CEP e as informações já aparecerem sem ao menos recarregar a página, certo?

Da nossa forma inicial, ou seja, recarregando a página, não entregaríamos essa experiência para o usuário... E agora?

**Entendendo o AJAX**

Através do Javascript conseguimos enviar requisições para determinadas URLs e obter suas respostas sem que as páginas sejam recarregadas, em outras palavras, quando o usuário sair do campo do CEP ou clicar em um botão, a busca do mesmo será efetuada e as informações serão preenchidas para usuário sem que a página recarregue.

Esse método de requisição no Javascript é conhecido como [AJAX](https://developer.mozilla.org/pt-BR/docs/AJAX). Legal, mas como poderíamos aplicar esse tal de AJAX? Em um primeiro momento podemos pensar em simplesmente realizar o download de uma biblioteca contendo a funcionalidade que precisamos, talvez a mais famosa seja o [JQuery](https://jquery.com/" \t "_blank).

Porém não parece ser uma boa idéia incorporar uma biblioteca em nosso projeto apenas para realizar um AJAX, pois estaríamos utilizando cerca de 10% do que a mesma pode nos oferecer.

Então como devemos solucionar o problema sem criar uma dependência com terceiros? Criaremos nossa própria funcionalidade para buscar os CEPs para atingir o resultado desejado.

**Pegando o formulário HTML**

Nosso primeiro passo será de fato pegar o nosso formulário HTML para aplicarmos o AJAX. Para este exemplo farei uso [deste formulário](https://github.com/mahenrique94/ajax-com-js-puro/archive/master.zip) que está no meu Github, ou seja, se você quiser seguir o passo a passo, pode baixar também! ;)



Com o HTML pronto, precisamos começar a implementação da busca de CEP conforme nosso cliente havia solicitado, para tal recurso iremos utilizar o [web service](https://pt.wikipedia.org/wiki/Web_service) do [viacep](https://viacep.com.br/" \t "_blank), um serviço gratuito na web que nos permite recuperar informações referente a determinados CEP's.

**Criando requisições com XMLHttpRequest**

O Javascript nativo possui um objeto responsável por realizar requisições AJAX conhecido como [XMLHttpRequest](https://developer.mozilla.org/pt-BR/docs/Web/API/XMLHttpRequest" \t "_blank), ou seja, por meio dele podemos recuperar dados de uma determinada URL. Apesar do nome, ele pode ser usado para recuperar qualquer tipo de dado e fornece suporte aos protocólos *HTTP, FILE e FTP*.

Vamos então criar uma instância do objeto para começarmos a utilizá-lo:

let xhr = new XMLHttpRequest();

Com o objeto criado, vamos utilizar o método [open](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/open) para começarmos a criar a nossa requisição:

xhr.open('GET', url, true);

O método open recebe três parâmetros, sendo eles:

1. O [verbo HTTP](https://www.alura.com.br/artigos/metodos_de_requisicao_do_http) a ser utilizado para realizar a requisição, o mesmo deve seguir o [padrão REST](https://www.caelum.com.br/apostila-vraptor-hibernate/rest/#11-1-o-que-e-rest).
2. A URL que pretendemos obter os dados.
3. Um argumento booleano informando se a requisição deve ser assíncrona ou síncrona.

Agora precisamos implementar a propriedade [onreadystatechange](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/onreadystatechange" \t "_blank) que é disparada sempre que nossa requisição sofre alguma alteração durante seu processamento:

xhr.onreadystatechange = function() {

}

Repare que passamos uma função de *callback*, ou seja, uma função que sempre será executada assim que a requisição sofrer alguma alteração. Quando implementamos essa função precisamos tomar alguns cuidados, o primeiro é verificar se a mesma foi finalizada, mas como fazer essa verificação? Fazemos isso através do [xhr.readyState](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/readyState" \t "_blank):

xhr.onreadystatechange = function() {

if (xhr.readyState == 4)

{ *//requisição finalizada }*

}

O código 4 nos informa que a requisição foi finalizada, porém ainda não é suficiente, afinal, f**inalizada não significa que foi bem sucedida**, portanto, é necessário verificar se ela foi realizada com sucesso também! Portanto, faremos uso do [xhr.status](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/status" \t "_blank) pra isso:

xhr.onreadystatechange = function() {

if (xhr.readyState == 4) {

if (xhr.status = 200)

*// faz alguma coisa*

} }

O código 200 nos informa que a requisição obteve sucesso durante seu processamento. Isso significa que agora podemos acessar a resposta que o servidor nos enviou e realizar alguma função desejada.

**Preenchendo os campos no formulário**

Até o momento já devemos ser capazes de obter as informações do nosso serviço de CEP, começaremos então a criar uma função que será invocada a cada clique em um botão:

function buscaCep() {

}

O primeiro passo será pegar a referência do campo CEP para obtermos seu valor:

let inputCep = document.querySelector('input[name=cep]');

let cep = inputCep.value;

Feito isso já podemos começar construir nossa URL:

let url = 'http://viacep.com.br/ws/' + cep + '/json';

Com nossa URL pronta, podemos começar a implementar o XMLHttpRequest conforme visto anteriormente:

let xhr = new XMLHttpRequest();

xhr.open('GET', url, true);

xhr.onreadystatechange = function() {

if (xhr.readyState == 4) {

if (xhr.status = 200)

console.log(xhr.responseText);

} }

xhr.send(); }

Tudo pronto, certo? Entretanto nosso web service exige que o CEP venha apenas com números, em outras palavras, se deixarmos apenas os dados que estamos pegando do campo referente ao CEP, podemos enviar, por exemplo, um CEP com o seguinte formato "12345-123", que é uma forma comum em formulários na web.

Caso isso aconteça nossa requisição irá falhar e não conseguiremos obter as informações... Como podemos resolver o problema? Precisamos de fato retirar a formatação do CEP antes de enviá-lo da seguinte maneira:

*// código omitido*

let cep = inputCep.value.replace('-', ''); *// código omitido*

Realizamos um replace para retirar o - do CEP, agora para testar nosso serviço podemos informar o CEP 13845-373 em nosso formulário e clicar no botão, onde iremos ver a resposta do servidor:

{

"cep" : "13845-373",

"complement" : "",

"logradouro" : "Rua Caiapós",

"bairro" : "Jardim Igaçaba",

"localidade" : "Mogi Guaçu",

"uf" : "SP",

"unidade" : "",

"ibge" : "3530706",

"gia" : "4558" }

Como podemos ver, através do [responseText](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/responseText" \t "_blank) conseguimos obter a resposta do nosso servidor, porém o JSON está em formato de string, sendo assim, precisamos convertê-lo para um objeto JSON em Javascript, podemos fazer isso através do [JSON.parse](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/JSON/parse" \t "_blank):

*// código omitido*

console.log(JSON.parse(xhr.responseText)); *// código omitido*

Ao mandar nosso formulário buscar o CEP novamente, iremos notar a mudança da saída em nosso console que agora passou a ser um objeto e não uma string:

Object {cep : "13845-373", logradouro : "Rua Caiapós", complemento : "", bairro : "Jardim Igaçaba", localidade : "Mogi Guaçu"...}

Para finalizar, precisamos pegar os dados do objeto e passar para nossos inputs, para isso vamos criar uma função que recebe o JSON como parâmetro e popula nossos campos:

function preencheCampos(json) {

document.querySelector('input[name=endereco]').value = json.logradouro;

document.querySelector('input[name=bairro]').value = json.bairro;

document.querySelector('input[name=complemento]').value = json.complemento;

document.querySelector('input[name=cidade]').value = json.localidade;

document.querySelector('input[name=estado]').value = json.uf; }

Nossa função completa fica da seguinte maneira:

function buscaCep() {

let inputCep = document.querySelector('input\[name=cep\]');

let cep = inputCep.value.replace('-', '');

let url = 'http://viacep.com.br/ws/' + cep + '/json';

let xhr = new XMLHttpRequest(); xhr.open('GET', url, true);

xhr.onreadystatechange = function() {

if (xhr.readyState == 4) {

if (xhr.status = 200) preencheCampos(JSON.parse(xhr.responseText)); } }

xhr.send(); }

Ao realizarmos a busca novamente iremos notar que nossos inputs serão preenchidos com a resposta do servidor:



Veja que agora já somos capazes de realizar requisições AJAX do tipo GET com Javascript puro. Dessa forma evitamos a necessidade de depender de uma biblioteca para um simples recurso e conseguimos entregar a funcionalidade nova para nosso cliente.