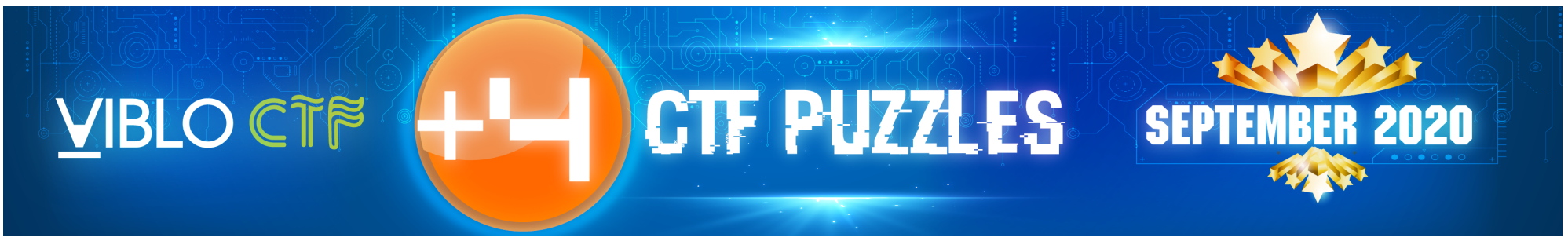

Pham Bao Khanh @khanh.pb25 Follow
894 25 15

Published Oct 23rd, 2017 4:35 AM - 9 min read
24.8K 3 5

Hướng dẫn về cách dùng hàm $.ajax() của JQuery

Mở đầu

Ajax đã nhanh chóng trở thành một phần quan trọng và phổ biến trong việc phát triển web và là một trong các mô hình thành công nhất từ trước đến giờ. Ajax có một số phương thức ngắn gọn được dùng phổ biến như: `$.get()`, `$.post()` và `$.load()`. Chúng là những phương thức rất tiện lợi được sử dụng để tạo các request Ajax chỉ với vài dòng code.

Nhưng đôi khi chúng ta cần nhiều sự kiểm soát hơn cho mỗi lần ta sử dụng Ajax. Ví dụ, chúng ta muốn chỉ rõ những hành động sẽ được thực hiện, sẽ xử lý như thế nào trong trường hợp một request Ajax thất bại hay chúng ta cần thực hiện một request Ajax nhưng chỉ cần kết quả của nó nếu kết quả này được lấy về trong đúng một khoảng thời gian do chúng ta quy định. Trong các trường hợp như thế, chúng ta có thể trông cậy vào một hàm khác được cung cấp bởi JQuery, đó là `$.ajax()`.

Hàm `$.ajax()`

Hàm `$.ajax()` của JQuery được sử dụng để thực hiện các request HTTP bất đồng bộ. Nó đã được thêm vào thư viện từ rất lâu, tồn tại từ phiên bản 1.0. Ba hàm `$.get()`, `$.post()` và `$.load()` được đề cập ở trên có thể coi là một hàm `$.ajax()` với những thiết lập có sẵn. Sau đây là cú pháp tổng quát của hàm `$.ajax()`:

```
$.ajax(url[, options])
$.ajax([options])
```

Tham số url là một chuỗi chứa URL mà bạn muốn sử dụng AJAX để thực hiện request, trong khi đó tham số options là một object thuần chứa các thiết lập cho request AJAX đó.

Ở dạng đầu tiên, phương thức này thực hiện một request AJAX sử dụng tham số **url** và các cài đặt được chỉ định ở **options**. Ở dạng thứ hai, URL được chỉ định trong tham số **options**, hoặc có thể được lược bỏ trong trường hợp request này được gửi đến chính đường dẫn của trang hiện tại.

Danh sách các tham số option

Có rất nhiều các lựa chọn để bạn có thể thiết lập hàm `$.ajax()` hoạt động theo ý muốn của mình. Trong danh sách dưới đây, bạn có thể tìm các tên của option và mô tả của chúng theo thứ tự bản chữ cái:

- **accepts**: Nội dung được gửi trong request header giúp server biết được kiểu response server sẽ chấp nhận khi trả về.
- **async**: Thiết lập giá trị `false` để thực hiện một request đồng bộ.
- **beforeSend**: Một hàm pre-request gọi lại có thể dùng để điều chỉnh object `jqXHR` trước khi nó được gửi.
- **cache**: Thiết lập giá trị `false` để buộc browser không lưu cache các trang được request.
- **complete**: Một hàm được thực thi khi request kết thúc (sau khi hàm gọi lại `success` và `error` được thực thi).
- **contents**: Một object của string hoặc REGEX dùng để xác định xem JQuery sẽ phân tích response như thế nào.
- **contentType**: Kiểu nội dung của dữ liệu được gửi lên server.
- **context**: Một object được dùng làm ngữ cảnh (`this`) của tất cả các hàm gọi lại liên quan đến Ajax.
- **crossDomain**: Thiết lập thuộc tính này là `true` để buộc thực hiện request chéo giữa các domain (như là JSONP) trên cùng một domain.
- **data**: Dữ liệu được gửi lên server khi thực thi một request Ajax.
- **dataFilter**: Một hàm được dùng để xử lý các dữ liệu response thuần của một XMLHttpRequest.
- **dataType**: Kiểu của dữ liệu mong muốn được trả về từ server.
- **error**: Một hàm sẽ được gọi khi request fails.
- **global**: Dùng để thiết lập xem có gọi các hàm xử lý sự kiện Ajax toàn cục cho request này hay không.
- **headers**: Một object để viết thêm vào các header gửi lên server.
- **ifModified**: Thiết lập giá trị này là `true` nếu bạn muốn buộc JQuery nhận diện môi trường hiện tại là "**local**".
- **jsonp**: Một chuỗi dùng để override tên hàm gọi lại trong một request JSONP.
- **jsonpCallback**: Chỉ định tên hàm gọi lại cho một request JSONP.
- **mimeType**: Một chuỗi chỉ định kiểu mime dùng để override lại kiểu mime của XHR.
- **password**: Mật khẩu được sử dụng với XMLHttpRequest cho response của một request yêu cầu xác thực truy nhập HTTP.
- **processData**: Set giá trị này là `false` nếu bạn không muốn dữ liệu được truyền vào thiết lập `data` sẽ được xử lý và biến thành một query kiểu chuỗi.
- **scriptCharset**: Thiết lập thuộc tính charset của một thẻ script dùng cho một request nhưng chỉ áp dụng khi **transport script** (ví dụ: request chéo giữa các domain với jsonp) được sử dụng.
- **statusCode**: Một object chứa các mã HTTP ở dạng số và các hàm được gọi khi response trả về có chứa một mã tương ứng.
- **success**: Một hàm được gọi khi request thành công.
- **timeout**: Số được thiết lập chỉ định thời gian hết hạn cho một request.
- **traditional**: Thiết lập giá trị `true` nếu bạn mong muốn param được serialize theo kiểu truyền thống.

- **xhr**: Một hàm gọi lại dùng để tạo một object XMLHttpRequest.
- **xhrFields**: Một object các key-value được thiết lập cho object XHR native.

Thực tế sử dụng của một vài thiết lập

Ở phần này chúng ta sẽ được xem hàm `$.ajax()` và một vài thiết lập hoạt động như thế nào.

Ví dụ đầu tiên của `$.ajax()`

Chúng ta sẽ bắt đầu với một demo đơn giản so sánh giữa `$.load()` và `$.ajax()`:

```
$('#main-menu a').click(function(event) {
   event.preventDefault();

   $('#main').load(this.href + ' #main *', function(responseText, status) {
      if (status === 'success') {
         $('#notification-bar').text('The page has been successfully loaded');
      } else {
         $('#notification-bar').text('An error occurred');
      }
   });
});
```

Cập nhật đoạn code trên với hàm `$.ajax()`, ta sẽ được đoạn code sau:

```
$('#main-menu a').click(function(event) {
   event.preventDefault();

   $.ajax(this.href, {
      success: function(data) {
         $('#main').html($(data).find('#main *'));
         $('#notification-bar').text('The page has been successfully loaded');
      },
      error: function() {
         $('#notification-bar').text('An error occurred');
      }
   });
});
```

Ở đây bạn có thể thấy rằng dạng thứ nhất của `$.ajax()` được sử dụng, URL để gửi request được thiết lập với tham số đầu tiên và sau đó là object các thiết lập. Ở đây có 2 thiết lập được sử dụng đến trong danh sách trên đó là `success` và `error` để chỉ định sẽ làm gì trong trường hợp request thành công hay thát bại.

Lấy về dữ liệu của một cuộc nói chuyện từ *Joind.in* sử dụng `$.ajax()`

Ở ví dụ thứ hai, ta sẽ bàn luận về việc tạo một request JSONP để lấy về thông tin từ một service của **Joind.in**. Đây là một website nơi những đã tham gia một sự kiện nào có thể để lại feedback cho sự kiện đó và ban tổ chức sự kiện. Đoạn code dưới dây sẽ sử dụng `$.ajax()` để lấy về **title** và **description** của sự kiện cuộc trò chuyện về *"Front-end thời hiện đại dưới cái nhìn của một developer PHP"*.

```
$.ajax({
   url: 'http://api.joind.in/v2.1/talks/10889',
   data: {
      format: 'json'
   },
   error: function() {
      $('#info').html('<p>An error has occurred</p>');
   },
   dataType: 'jsonp',
   success: function(data) {
      var $title = $('<h1>').text(data.talks[0].talk_title);
```



```
         .append($description);
   },
   type: 'GET'
});
```

Ở đoạn code trên, một vài thuộc tính khác đã được sử dụng thêm. Đầu tiên, bạn có thể thấy rằng dạng thứ hai của `$.ajax()` được sử dụng, giúp bạn có thể chỉ định URL mà request này gửi đến thông qua thuộc tính (`url`). Vì API của Joind.id chấp nhận các request JSONP, nên trong đoạn code trên, kiểu request được thiết lập thông qua thuộc tính `dataType` . Sau đó, thuộc tính `data` được xử dụng để xác định kiểu của định dạng dữ liệu mà ta muốn lấy về từ server như được yêu cầu từ API. Thuộc tính `type` được set với giá trị `GET` để thực hiện một request GET.

Các bạn có thể xem thửu một demo trực tiếp của đoạn code trên được thực hiện trên JSfiddle.

Kết luận

Trong bài biết này, chúng ta đã bàn luận về một hàm AJAX vô cùng mạnh mẽ từ JQuery là `$.ajax()` . Nó cho phép bạn thực hiện các request AJAX với rất nhiều các thiết lập giúp bạn có thể điều khiển được request gửi lên server và cách mà response của nó được xử lý. Nhờ có hàm này mà giờ đây bạn đã có một công cụ để có thể thỏa mãn rất nhiều nhu cầu trong project của bạn trong trường hợp các hàm `$.load()` , `$.post()` và `$.get()` đơn thuần không thể đáp ứng hết.

jQuery jQuery functions JavaScript

29 min read
2983 6 1 5

4 min read
2052 5 0 2

[Cách sử dụng Swagger để kiểm tra API (Laravel) (Phần 1)]

Lê Bảo Ngân
8 min read
1847 2 1 3

[Các hàm xử lý chuỗi trong PHP - phần 1]

Chu Thi Thuy Hang
7 min read
2776 2 2 10

More from Pham Bao Khanh

[Giới thiệu thuộc tính mới của Class trong Javascript: Trường Private và cách sử dụng nó]

Pham Bao Khanh
6 min read
1996 2 0 8

[5 Best Practices về cấu trúc khi làm việc với React]

Pham Bao Khanh
13 min read
2374 20 1 15

[Các best practices trong việc sử dụng những cú pháp Javascript thời hiện đại]

Pham Bao Khanh
20 min read
1144 10 0 11

[Giới thiệu về Axios - một HTTP Client dựa trên Promise của Javascript]

Pham Bao Khanh
12 min read
15533 12 7 17

Comments

Login to comment

Red Devil @manhtqb Oct 23rd, 2017 11:51 AM

Chào mọi người, mình có 1 câu hỏi mong mọi người giải đáp, có hơi ngô nghe mong ae bỏ quá 😃). Google có ra 1 vài câu trả lời nhưng nhờ ae giải thích kỹ hơn chút 😄

Dùng ajax thì tính tương tác tốt hơn, giấu được data và sử dụng cũng không quá khó. Tuy nhiên nếu sử dụng nhiều ajax thì có ảnh hưởng tới tốc độ web hay bảo mật không, ví dụ 1 request gửi = ajax thì có chậm hơn gửi = PHP ko chẳng hạn ? 😄

0 | Reply Share •••

Hoang vn @wiliamfeng Mar 20th, 2018 9:31 PM

tks bạn đã chia sẻ

0 | Reply Share •••

Phuong Dinh @pdinh Aug 14th, 2018 11:28 PM

minh moi lam quen voi django, minh da load du lieu thanh cong va hien thi ra bang. Minh muon su dung dataTable, tuy nhien minh k biet bat dau the nao van de cua minh giong y chang ban nay nhung minh van chua tim ra giai phap https://stackoverflow.com/questions/44866116/updating-datatable-with-django-model-objects ban co the huong dan 1 duong di cho minh dc k? minh chan thanh cam on

0 | Reply Share •••