MỘT SỐ KỸ THUẬT TĂNG TỐC ĐỘ XỬ LÝ DỮ LIỆU WEB VỚI NGÔN NGỮ LẬP TRÌNH JAVASCRIPT

NGUYỄN THỊ MỸ TRANG
Trường Cao đẳng Huế

Nhận bài ngày 09/5/2025. Sửa chữa xong 13/5/2025. Duyệt đăng 17/5/2025.

Abstract

JavaScript is a scripting language executed in the browser, enabling client-side processing. Since JavaScript code runs directly on the client machine without sending data to the server, it reduces dependency on server response time and saves bandwidth, thereby improving website responsiveness. This article presents several techniques to optimize data processing speed using JavaScript, with a focus on jQuery a lightweight JavaScript library that helps simplify common programming tasks by reducing code complexity and length.

Keywords: *JavaScript, computer science, website programming, HTML, DOM, jQuery.*

1. Đặt vấn đề

Trong bối cảnh khoa học công nghệ phát triển, yêu cầu cần có công cụ để quảng bá, giới thiệu hoạt động, hình ảnh của các cơ quan, tổ chức, công nghệ web hiện nay đang là xu thế tất yếu. Một hệ thống website đáp ứng yêu cầu nhanh, đẹp và tương tác tốt với người sử dụng được đặt lên hàng đầu. Để xây dựng một website, người lập trình web sử dụng HTML để định nghĩa nội dung trang web, CSS xác định cách hiển thị và JavaScript để lập trình hành vi của trang web. Vậy làm thế nào để website thực thi nhanh, đáp ứng các yêu cầu của người dùng, đó là vấn đề người lập trình web cần chú trọng.

2. Nội dung nghiên cứu

2.1. Vai trò của ngôn ngữ Javascript trong trang web

Javascript (JS) là ngôn ngữ lập trình được đưa ra vào năm 1995 bởi Brenden Eich. JS là ngôn ngữ lập trình kịch bản được thực thi trên trình duyệt và thực hiện các xử lý ngay tại máy trạm (client-side). Ban đầu, JS chỉ được hỗ trợ bởi trình duyệt Netscape, nhưng sau này JS đã được quy chuẩn và các trình duyệt khác đã cung cấp công nghệ để xử lý JS. JS có thể tương tác với các phần tử HTML, tạo các hiệu ứng, kiểm tra tính hợp lệ của dữ liệu trên trang web [3]. JS là ngôn ngữ lập trình HTML và Web. JS là một trong 3 ngôn ngữ mà tất cả những người phát triển web phải học: HTML định nghĩa nội dung trang web; CSS xác định cách hiển thị của trang web; JS để để lập trình hành vi của trang web. JS thường được nhúng trực tiếp vào các trang HTML. Người sử dụng có thể sử dụng JS mà không phải mua bản quyền.

JS có thể thực hiện ở các hình thức [5], [7]: Cung cấp cho người thiết kế HTML một công cụ lập trình. Tác giả của HTML thường không phải là người lập trình, nhưng JS là một ngôn ngữ lập trình kịch bản với cấu trúc rất đơn giản. Hầu hết mọi người đều có thể đưa một đoạn mã nhỏ vào trong trang web của họ. Có thể đưa một văn bản vào trang HTML. Có thể tác động đến sự kiện. JS có thể thiết lập để xử lý khi sự kiện gì đó xảy ra, như khi trang web đã nạp xong hay khi một người sử dụng kích vào một thành phần của HTML. Có thể đọc và ghi các thành phần HTML. JS có thể đọc và thay đổi nội dung của một thành phần HTML. Có thể được sử dụng xác nhận tính hợp lệ của dữ liệu. JS có thể được sử dụng để xác nhận dữ liệu hợp lệ trước khi được gửi đến máy chủ. Điều này tiết kiệm thời gian xử lý của máy chủ (dữ liệu

Email: mytrangsp@gmail.com

được xử lý từ máy trạm, sau đó mới gửi lên máy chủ). Có thể được sử dụng để tạo cookies. JS có thể được sử dụng để lưu trữ và truy hồi thông tin trên máy tính của người sử dụng.

2.2. Một số kỹ thuật tăng tốc độ xử lý của website với JavaScript

2.2.1. Vị trí đặt mã JS trong HTML

Đoạn mã JS có thể được đặt trong phần <body> hoặc phần <head> của trang HTML. Việc đặt thẻ <script> trong phần <head> hay trong phần <body> phụ thuộc vào trang. Nếu trang web của chúng ta tham chiếu đến một số file JS có dung lượng lớn, mất thời gian để nạp, thì tốt hơn hết nó được định nghĩa vào cuối của phần <body>. Vì khi trình duyệt bắt đầu tải trang, việc tải các đoạn mã JS không cản trở, làm chậm trễ đến sự hiển thị của trang. Chúng ta có thể định nghĩa các file JS trong phần <head> chỉ khi chúng không ảnh hưởng đến việc thực hiện của trang. Các đoạn mã JS nên được định nghĩa cuối cùng khi toàn bộ trang đã được nạp.

2.2.2. Thuộc tính async và defer

Để khắc phục tình trạng mã JS có thể làm ảnh hưởng đến quá trình nạp trang web, có một số thuộc tính như async và defer mà hầu hết các trình duyệt hiện nay đều hỗ trợ để tăng tốc độ tải trang web. Đối với thuộc tính async thì các phiên bản IE10 trở về sau mới hỗ trợ. Chúng ta có thể sử dụng các thuộc tính này bên trong thẻ <script>: <script src="…/test.js" async></script>. Các đoạn mã JS có khai báo thuộc tính async được xử lý một cách không đồng bộ mà không trở ngại đến trình duyệt tải trang. Khi chúng ta tải JS theo cách thông thường, trình duyệt sẽ đọc mã trang web theo thứ tự từ trên xuống dưới. Nếu gặp đoạn mã JS thì trình duyệt sẽ dừng việc nạp HTML và xử lý mã JS cho đến lúc kết thúc mới nạp tiếp HTML, đó là lý do vì sao khi chúng ta kiểm tra tốc độ website trên PageSpeed của Google thì phát hiện lỗi "*Loại bỏ JS và CSS chặn hiển thị nội dung trong màn hình đầu tiên*". Nếu chúng ta sử dụng thuộc tính async, khi trình duyệt gặp đoạn mã JS thì HTML bên dưới vẫn tiếp tục tải song song với việc tải mã JS. Khi mã JS đã được tải xong thì đoạn mã JS này được thực thi và HTML sẽ tạm ngừng cho đến khi JS thực hiện xong. Tuy nhiên, nếu có quá nhiều đoạn mã JS, mà mỗi đoạn mã được xử lý không đồng bộ và tại cùng một thời gian, thì có thể dẫn đến việc đoạn mã thứ hai được biên dịch hoàn thành trước đoạn mã thứ nhất. Điều này có thể phát sinh một số lỗi nếu các đoạn mã này phụ thuộc lẫn nhau.

Ví dụ, khi đang làm việc với các framework trên máy trạm, như Angular, mã JS sử dụng các thành phần Angular thuộc thư viện AngularJS. Trong trường hợp này, nếu các file JS của chúng ta nạp trước thư viện AngularJS mà chúng phụ thuộc vào thì các đoạn mã JS sẽ bị lỗi. Để khắc phục tình trạng này, chúng ta sử dụng thuộc tính defer để xử lý các đoạn mã JS theo tuần tự. Chúng ta sử dụng thuộc tính defer như sau:

<script src="…/test1.js" defer> </script>.

<script src="…/test2.js" defer> </script>.

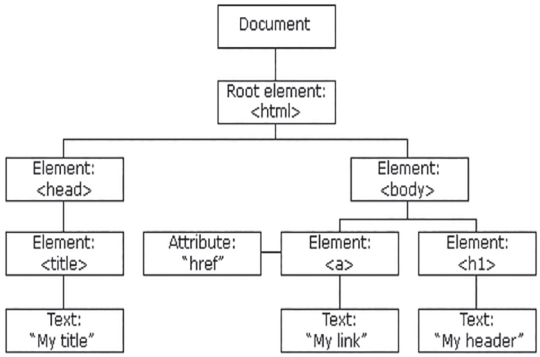
Về cơ bản, cả hai thuộc tính async và defer đều có cách thức tải giống nhau, đều tải đồng thời các thẻ HTML khác, chỉ khác là thời gian thực thi. JS có sử dụng thuộc tính async sẽ được thực thi ngay khi tải xong (dừng tải HTML), còn JS có sử dụng thuộc tính defer sẽ được thực thi sau khi nạp xong HTML.

Tóm lại, chúng ta nên dùng thuộc tính async và defer khi: Nếu đoạn mã JS của chúng ta độc lập và không phụ thuộc vào bất kỳ đoạn mã JS nào khác thì dùng thuộc tính async. Nếu đoạn mã JS của chúng ta phụ thuộc vào một script khác hoặc một script khác phụ thuộc nó thì nên sử dụng thuộc tính defer. Nếu script của chúng ta nhỏ, chỉ có vài dòng code thì xuất ngay các đoạn mã vào trong thẻ <script>, không cần tạo file JS độc lập và không dùng 2 thuộc tính đó.

2.2.3. Sử dụng jQuery

a. Mô hình đối tượng DOM

Trước khi tìm hiểm jQuery, chúng ta tìm hiểu sơ lược về DOM. Khi trang web được nạp, trình duyệt tạo một mô hình đối tượng tài liệu của trang web, hay còn gọi là DOM (Document Object Model). Mô hình HTML DOM được xây dựng như một cây đối tượng (Hình 1). Với HTML DOM, Javascript có thể truy cập và thay đổi tất cả các phần tử (element) của một tài liệu HTML.



Hình 1: Mô hình đối tượng DOM

b. Giới thiệu jQuery

jQuery là một thư viện của JavaScript, được tạo bởi John Resig vào năm 2006. jQuery giúp viết mã JS nhanh hơn, gọn hơn, đơn giản hóa các tác vụ đa dạng với việc viết ít code hơn. jQuery làm đơn giản hóa việc truyền tải HTML, xử lý sự kiện, tạo hiệu ứng động và tương tác Ajax. Một số tính năng quan trọng được hỗ trợ bởi jQuery: Thao tác DOM: jQuery giúp chúng ta duyệt, lựa chọn các phần tử DOM để thao tác một cách dễ dàng; Xử lý sự kiện: jQuery giúp tương tác với người dùng tốt hơn bằng việc xử lý các sự kiện đa dạng; Hỗ trợ AJAX: jQuery giúp người lập trình web phát triển một site giàu tính năng và phản hồi tốt bởi sử dụng công nghệ AJAX; Hiệu ứng: jQuery đi kèm với nhiều hiệu ứng đa dạng và đẹp mắt; Gọn nhẹ: jQuery là thư viện gọn nhẹ, có kích cỡ khoảng 19KB (đã nén); Tính tương thích: jQuery được hỗ trợ bởi hầu hết các trình duyệt hiện đại; Cập nhật và hỗ trợ các công nghệ mới nhất.

jQuery là một thư viện JavaScript nên để sử dụng, chúng ta phải chèn thư viện này vào trang web. Ưu điểm lớn nhất của việc sử dụng jQuery từ máy chủ của Google hoặc Microsoft là nhiều người dùng đã thật sự tải jQuery từ Google hoặc Microsoft về máy khi viếng thăm các trang web khác, do đó thời gian nạp trang nhanh hơn, tiết kiệm băng thông cho trang web. Ngoài ra, khi người sử dụng yêu cầu một file từ jQuery, nó sẽ được phục vụ từ một server gần nhất, do đó thời gian nạp cũng nhanh.

c. jQuery Selector

Một trong những tính năng mạnh mẽ nhất của jQuery là khả năng chọn các thành phần trong DOM một cách dễ dàng. Các jQuery Selector cho phép chúng ta chọn và xử lý các phần tử HTML. Các jQuery Selector được sử dụng để tìm/chọn các phần tử HTML dựa trên ID, lớp, kiểu, thuộc tính, giá trị của thuộc tính… jQuery Selector dựa trên các CSS Selector và các Selector riêng của jQuery. Thư viện jQuery hỗ trợ gần như toàn bộ các CSS Selector chuẩn, từ CSS1 cho đến CSS3. Do đó, người lập trình web không phải lo lắng về việc trình duyệt đó có hỗ trợ những Selector mới hay không, miễn là trình duyệt đó có bật JavaScript. Tất cả các Selector trong jQuery đều bắt đầu với $ (bảng 2).

Selector	Ví dụ	Giải thích
*	$(“*”)	Tất cả các phần tử
:first	$(“p:first”)	Thẻ <p> đầu tiên
:last	$(“p:last”)	Thẻ <p> cuối cùng
:even	$(“tr:even”)	Tất cả dòng chẵn trong trang
:odd	$(“tr:odd”)	Tất cả dòng lẻ trong trang
:first-child	$(“p:first-child”)	Tất cả thẻ p là con đầu tiên của một thẻ nào đó
:last-child	$(“p:last-child”)	Tất cả thẻ p là con cuối cùng của một thẻ nào đó
:first-of-type	$(“p:first-of-type”)	Tất cả thẻ p đầu tiên là con của một thẻ nào đó
:last-of-type	$(“p:last-of-type”)	Tất cả thẻ p cuối cùng là con của một thẻ nào đó

Bảng 2: Một số Selector riêng của jQuery

Lưu ý: jQuery đánh chỉ mục bắt đầu từ 0.

Khi làm việc với Form, những Selector của jQuery giúp chúng ta tiết kiệm thời gian để chọn chỉ những thành phần nào mình muốn. Bảng sau là những Selector của jQuery để làm việc với Form (bảng 3).

Selector	Thành phần được chọn
:text, :checkbox, :radio, :image, :submit, :reset, :password, :file	Phần tử Input với thuộc tính có cùng tên với Selector. Ví dụ: Selector :text sẽ chọn phần tử <input type='text' />
:input	Input, textarea, select và các button
:button	Các phần tử nút và input với thuộc tính type="button"
:enable	Phần tử form được bật
:disable	Phần tử form bị tắt
:checked	Nút radio hoặc hộp kiểm được đánh dấu

Bảng 3: Một số Selector sử dụng cho Form

Cũng giống như các Selector khác, Selector dùng cho Form được sử dụng để chọn một đối tượng cụ thể. Ví dụ: Chọn tất cả các nút radio được đánh dấu với lệnh $(':radio:checked'). Hoặc chọn tất cả các trường nhập mật khẩu và trường nhập dữ liệu bị tắt với lệnh $(':password, :text:disable'). Với jQuery Selector, chúng ta vẫn sử dụng nguyên tắc của CSS để chọn các phần tử cần chọn.

d. Phương thức duyệt DOM trong jQuery

Các jQuery Selector ở trên cho phép chúng ta chọn một tập hợp các phần tử khi chúng ta di chuyển ngang qua hoặc dọc xuống cây DOM và chọn lọc kết quả. Bảng sau cung cấp một số phương thức để duyệt DOM.Các jQuery Selector ở trên cho phép chúng ta chọn một tập hợp các phần tử khi chúng ta di chuyển ngang qua hoặc dọc xuống cây DOM và chọn lọc kết quả. Bảng sau cung cấp một số phương thức để duyệt DOM *(bảng 4)*.

Phương thức	Chức năng
add()	Thêm phần tử vào nhóm các phần tử đang tồn tại (để thêm cùng định dạng CSS, …).
children()	Trả về tất cả các con trực tiếp của phần tử chọn.
parent()	Trả về phần tử cha trực tiếp của phần tử chọn.
parents()	Trả về tất cả các phần tử tổ tiên của phần tử chọn.
closest()	Trả về tổ tiên đầu tiên của phần tử chọn (cha của cha)
next()	Trả về phần tử anh em kế tiếp của phần tử chọn
siblings()	Trả về tất cả phần tử anh em của phần tử chọn
prev()	Trả về phần tử anh em đằng trước của phần tử chọn.

Bảng 4: Một số phương thức duyệt DOM

e. Phương thức sự kiện

Các phương thức sự kiện (Event methods) gán một hàm với một trình quản lý sự kiện cho các phần tử được chọn. Các sự kiện làm cho trang web sinh động và tương tác tốt với người sử dụng. jQuery nâng cao và mở rộng cơ chế quản lý sự kiện cơ bản giúp chúng ta viết code ngắn gọn hơn, tiết kiệm thời gian hơn và hỗ trợ lập trình mạnh mẽ hơn. Phương thức sự kiện $(document).ready() trong jQuery hoạt động tương đương với sự kiện onload trong JavaScript. Thực tế, hai phương thức sự kiện này có tác dụng giống nhau nhưng chúng lại kích hoạt tác vụ ở những thời điểm khác nhau. Sự kiện window.onload được kích hoạt khi trình duyệt đã hoàn toàn tải xong tài liệu, có nghĩa là mọi phần tử trên trang đã sẵn sàng để được thao tác bởi JS. Điều này giúp người lập trình không cần quan tâm đến trật tự tải dữ liệu. Phương thức sự kiện $(document).ready() được kích hoạt khi DOM hoàn toàn sẵn sàng để sử dụng, nghĩa là mọi thành phần có thể được truy cập bởi code của chúng ta, nhưng không nhất thiết là tài liệu liên quan đã được tải xong. Ngay sau khi HTML được tải và chuyển qua cây DOM, code có thể được thực thi.

Lưu ý: Để đảm bảo trang web vẫn có định dạng trước khi code JS được thực hiện, người ta thường đặt <link rel="stylesheet"> ở trước thẻ <script> trong phần <head> của tài liệu.

Ví dụ, chúng ta có một trang thư viện ảnh, trang đó gồm nhiều hình ảnh có dung lượng lớn mà

chúng ta có thể ẩn, hiện, di chuyển hoặc thao tác với jQuery. Nếu chúng ta thiết lập giao diện sử dụng sự kiện onload thì người dùng sẽ phải đợi cho đến khi mọi hình ảnh đã được tải xong mới có thể sử dụng trang web. Tuy nhiên, nếu chúng ta sử dụng $(document).ready(), thì giao diện sẵn sàng để sử dụng cho dù các hình ảnh chưa được tải xong.

Thực tế, phương thức sự kiện $(document).ready() được sử dụng nhiều hơn sự kiện onload. Tuy nhiên, chúng ta cần lưu ý khi sử dụng $(document).ready() một số tập tin hỗ trợ có thể chưa được tải, cho nên một số thuộc tính như độ cao, chiều rộng của ảnh có thể chưa có sẵn trong lúc này. $(document).ready() thực chất là gọi phương thức ready() cho một đối tượng jQuery mà chúng ta đã tạo ra từ phần tử DOM.

Trong thực tế, chúng ta có nhu cầu sử dụng nhiều hơn một thư viện JS trên cùng một trang. Vì nhiều thư viện cùng sử dụng ký hiệu nhận dạng $ do nó ngắn và thuận tiện nên chúng ta phải tránh xảy ra xung đột giữa các thư viện cùng sử dụng ký hiệu này. jQuery cung cấp phương thức noConflict() để trả ký hiệu nhận dạng về cho các thư viện khác.

Ví dụ:

```
<script src="prototype.js" type="text/javascript"></script>
<script src="jquery.js" type="text/javascript"></script>
<script type="text/javascript">
jQuery.noConflict();
</script>
<script src="myscript.js" type="text/javascript"></script>
```

Đầu tiên thư viện Prototype được gọi, đây là một thư viện JS, sau đó jQuery được gọi và cũng sử dụng ký pháp $. Phương thức noConflict() được gọi để giải phóng $, chuyển quyền điều khiển cho thư viện được gọi đầu tiên là thư viện Prototype. Bây giờ, đoạn mã chúng ta có thể sử dụng cả hai thư viện, nhưng bất kỳ lúc nào chúng ta muốn sử dụng một phương thức jQuery, chúng ta cần phải sử dụng jQuery thay vì dấu $ làm ký hiệu nhận dạng. Nếu muốn sử dụng $ trong lệnh jQuery ở trường hợp này mà không bị xung đột, chúng ta có thể khai báo như sau:

```
jQuery(document).ready(function($) {
//trong này, chúng ta có thể sử dụng $ bình thường.
});
```

Hoặc sử dụng cách viết tắt:

```
jQuery(function($) {//đoạn mã sử dụng $}); (bảng 5):
```

Phương thức	Chức năng	Ví dụ
Bind()	Gán sự kiện vào cho các phần tử HTML	$("p").bind("click",function(){ alert("Đoạn văn đã được kích chọn"); });
Blur()	Gán sự kiện làm mất tiêu điểm trong vùng nhập dữ liệu.	$("input").blur(function(){ Alert("Đây là trường bị mất tiêu điểm"); });
Change()	Gán sự kiện khi giá trị của một phần tử thay đổi.	$("input").change(function(){ alert("Văn bản đã bị thay đổi"); });
Click()	Gán sự kiện kích chuột.	$("p").click(function(){ alert("Đoạn văn này được kích chọn"); });

Bảng 5: Một số phương thức của jQuery thường dùng để quản lý sự kiện

f. Hiệu ứng

Các hiệu ứng động của jQuery sẽ làm cho trang web sinh động hơn. jQuery cho phép chúng ta điều khiển sự xuất hiện, di chuyển, tạo các hiệu ứng với các thành phần của trang web (bảng 6).

Phương thức	Chức năng	Ví dụ

Hide()	Ẩn các phần tử chọn.	$(document).ready(function(){
		$(".btn1").click(function(){
		$("p").hide();
		};
		$(".btn2").click(function(){
		$("p").show();
		};
		};

Show()	Hiển thị các phần tử chọn.	

Toggle()	Chuyển đổi giữa phương thức hide() và show().	$(document).ready(function(){
		$("button").click(function(){
		$("p").toggle();
		};
		};

Animate()	Tạo hiệu ứng hoạt hình với phần tử được chọn (thay đổi thuộc tính CSS như thay đổi độ cao, độ rộng, …)	$(document).ready(function(){
		$("#btn1").click(function(){
		$("#box").animate({height: "300px"}
		};
		$("#btn2").click(function(){
		$("#box").animate({height: "100px"}
		};
		};

Bảng 6: Một số phương thức của jQuery để tạo hiệu ứng

g. Cập nhật DOM

jQuery cũng cung cấp các phương thức để chúng ta thực hiện bổ sung, xóa, chỉnh sửa các phần tử trong DOMM.

3. Kết luận

JavaScript là ngôn ngữ lập trình web, tạo các hiệu ứng động, tương tác phổ biến trên trang web. Bài báo này cung cấp một số kỹ thuật nhằm tăng tốc độ xử lý web, đặc biệt là jQuery, sử dụng thư viện jQuery được xây dựng trên nền tảng JavaScript để xây dựng các hiệu ứng trên trang web một cách dễ dàng, thuận lợi hơn. jQuery làm đơn giản hóa việc truyền tải HTML, xử lý sự kiện, tạo hiệu ứng động và tương tác Ajax. Mỗi một kỹ thuật trình bày trong bài báo được tích rút kinh nghiệm từ thực tế, giúp người lập trình web làm chủ được các công nghệ mới.

Tài liệu tham khảo

[1] Andrea Chiarelli (2016), *Mastering JavaScript Object-Oriented Programming* (*Làm chủ lập trình hướng đối tượng JavaScript*), Packt Publishing, Ltd.

[2] Danny Goodman (2007), *JavaScript and DHTML Cookbook* (*Sách hướng dẫn JavaScript và DHTML*), O'Reilly Media, USA.

[3] Ovais Mehboob Ahmed Khan (2016), *JavaScript for .NET Developers* (*JavaScript cho người phát triển .NET*), Packt Publishing Ltd.

[4] Rehan Zaidi (2017), *JavaScript Essentials for SAP ABAP Developers: A Guide to Mobile and Desktop Application Development* (*JavaScript cơ bản cho nhà phát triển SAP ABAP: Hướng dẫn phát triển ứng dụng cho di động và máy tính để bàn*), Springer Science+Business Media New York.

[5] Russ Ferguson, Keith Cirkel (2017), *JavaScript Recipes: A Problem-Solution Approach* (*JavaScript: Một cách tiếp cận vấn đề - Giải pháp*), Springer Science+Business Media New York.

[6] Stuart Langridge (2005), *DHTML Utopia Modern Web Design Using JavaScript & DOM* (*DHTML Utopia thiết kế Web hiện đại sử dụng JavaScript & DOM*), SitePoint Pty, Ltd.

[7] www.w3schools.com.