ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Lê Đình Thanh, Nguyễn Việt Anh

Giáo trình

PHÁT TRIỂN ỨNG DỤNG WEB

Hà Nội, 2018

MỤC LỤC

BẢNG CÁC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ tiếng Việt	Thuật ngữ tiếng Anh	Viết tắt
An ninh tầng giao vận	Transport Layer Security	TLS
Ánh xạ đối tượng-quan hệ	Object-Relational Mapping	ORM
Bảng định dạng	Cascading Style Sheet	CSS
Bên khách	Client-side	
Bên phục vụ	Server-side	
Biểu diễn mã phần trăm	Percent-encoding	
Biểu diễn mã URL	URL encoding	
Bộ tiền xử lý siêu văn bản PHP	PHP Hypertext Preprocessor	PHP
Buộc	Binding	
Câu lệnh chuẩn bị trước	Prepared statements	
Câu lệnh có tham số	Parameterized statements	
Chính sách cùng nguồn gốc	Same-origin policy	SOP
Chuỗi nguyên mẫu	Prototype chain	
Chuỗi truy vấn	Query string	
Chuyển trạng thái trình diễn	Representational State Transfer	REST
Cơ sở dữ liệu	Database	CSDL
Có trạng thái	Stateful	
CSS ngoài	External CSS	
CSS trong	Internal CSS	
Dãy ký tự thoát	Escape sequence	
Dẫn ống	Pipelining	
Đào hầm	Tunneled	
Đáp ứng HTTP	HTTP response	
Đệ trình	Submit	
Điểm đánh dấu	Bookmark	

Điều khiển truy cập	Access control	
Điều kiện hỗ trợ	Supports condition	
Định kiểu động	Dynamic typing	
Định kiểu không tường minh	Implicit typing	
Định nghĩa kiểu tài liệu	Document Type Definition	DTD
Định tuyến URL	URL routing	
Định vị tài nguyên đồng nhất	Uniform Resource Locator	URL
Đối tượng cha	Parent element	
Đối tượng con	Child element	
Đối tượng dữ liệu PHP	PHP Data Objects	PDO
Đối tượng/phần tử tài liệu	Document object/element	
Dựa trên nguyên mẫu	Prototype-based	
Giao diện cổng chung	Common Gateway Interface	CGI
Giao diện lập trình ứng dụng	Application Programming Interface	API
Giao thức truyền siêu văn bản	Hypertext Transfer Protocol	HTTP
Hiển thị theo dòng	Inline	
Hiển thị theo khối	Block	
Hướng kết nối	Connection-oriented	
JavaScript và XML không đồng bộ	Asynchronous JavaScript and XML	AJAX
Kết nối HTTP liên tục	HTTP persistent connection	
Khách-phục vụ	Client-server	
Kiến trúc đa tầng	Multitier architecture, n-tier	
Kiểu MINE	MINE type	
Kiểu nội dung	Content type	
Kiểu phương tiện	Media type	
Ký pháp đối tượng JavaScript	JavaScript Object Notation	JSON
Lớp giả	Pseudo class	
Lưu trữ ảo	Virtual hosting	

Lưu trữ web	Hosting	
Máy tìm kiếm	Search engine	
Mã trên cấu hình	Code over configuration/coding by convention	CoC
Mạng phân phối nội dung	Content Delivery Network	CDN
Mặt sau	Backend	
Mặt trước	Frontend	
Mô hình đối tượng tài liệu	Document Object Model	DOM
Mô hình đối tượng trình duyệt	Browser Object Model	BOM
Mô hình hộp	Box model	
Mở rộng JavaScript	JavaScript eXtension	JSX
Mở rộng mail đa mục đích	Multipurpose Internet Mail Extensions	MINE
MVC	Model-View-Controller	
Ngăn xếp web	Web stack	
Ngôn ngữ đánh dấu có thể mở rộng	eXtensible Markup Language	XML
Ngôn ngữ đánh dấu siêu văn bản	Hypertext Markup Language	HTML
Nội dung web	Web content	
Phần tử giả	Pseudo element	
Phát triển mặt sau	Back-end/server-side development	
Phát triển mặt trước	Front-end/client-side development	
Phi kết nối	Connectionless	
Phương thức HTTP	HTTP method	
Proxy chuyển tiếp	Forward proxy	
Proxy ngược	Reverse proxy	
Quản lý phiên	Session management	
Siêu liên kết	Hyperlink	

Tài nguyên mặc định	Default document	
Tài nguyên web	Web resource	
Tầng socket an toàn	Secure Socket Layer	SSL
Tầng logic nghiệp vụ	Business logic tier	
Tầng trình diễn	Presentation tier	
Tầng truy cập dữ liệu	Data access tier	
Tham chiếu ký tự	Character reference	
Thẻ HTML	HTML tag	
Thông dịch	Interpret	
Trang web	Web page	
Trình duyệt web	Web browser	
Trình khách web	Web client	
Trình phục vụ HTTP	HTTP server	
Trình phục vụ ứng dụng	Application server	
Trình phục vụ web	Web server	
Truy vấn phương tiện	Media query	
Ứng dụng web	Web application	
URL ngữ nghĩa	Semantic URL	
URL phi ngữ nghĩa	Non-semantic URL	
Viết lại URL	URL rewrite	
Web	World Wide Web	WWW
Web API	Web API	
Website	Website	
Xác thực	Authentication	
Yêu cầu HTTP	HTTP request	

LỜI NÓI ĐẦU

Với sự thịnh hành của Internet và web, phát triển ứng dụng web đã trở thành một nhánh quan trọng trong công nghiệp phần mềm. Nhu cầu nhân lực về phát triển ứng dụng web tại các doanh nghiệp luôn ở mức cao và không ngừng tăng. Vì vậy, kiến thức và kỹ năng về phát triển ứng dụng web là một hành trang cần thiết đối với tất cả sinh viên ngành công nghệ thông tin.

Giáo trình này được xuất bản nhằm trang bị cho sinh viên hiểu biết một cách toàn diện và có hệ thống các kiến thức cốt lõi liên quan phát triển ứng dụng web, nắm bắt và sử dụng tốt một số công cụ và kỹ thuật hiện đại trong phát triển ứng dụng web, có thể phát triển và triển khai ứng dụng web trong công nghiệp, cũng như dễ dàng nắm bắt và làm chủ được các công nghệ tạo lập web trong tương lai. Ngoài ra, sinh viên có thể tự tạo lập công cụ cho phát triển ứng dụng web.

Nội dung giáo trình bao trùm từ các kiến thức cơ bản tới các kỹ năng, kỹ thuật nâng cao, thực tiễn và chi tiết trong lĩnh vực phát triển ứng dụng web, trong đó kiến thức theo W3C là nền tảng, cốt lõi. Không những vậy, nội dung giáo trình có tính trung lập với các công nghệ và công cụ phát triển ứng dụng web, giúp sinh viên có thể sử dụng, chuyển đổi công nghệ và công cụ khi cần thiết. Chương 1 trình bày kiến thức tổng quan về hệ thống web, kiến trúc tổng thể, cơ chế hoạt động của ứng dụng web. Các chương từ 2 đến 4 trình bày đầy đủ và sâu sắc kiến thức về phát triển mặt trước (front-end development), tức phát triển mã nguồn chạy ở trình duyệt. Chương 5 giới thiệu một số công cụ hiện đại và phổ biến cho phát triển mặt trước. Các chương từ 6 đến 9 trình bày kiến thức căn bản và nâng cao về phát triển mặt sau (back-end development), tức phát triển mã nguồn chạy ở bên phục vụ. Chương 10 giới thiệu một thư viện cho phát triển mặt sau.

Sinh viên trước khi học theo giáo trình này cần có hiểu biết căn bản về lập trình thủ tục, lập trình hướng đối tượng, cơ sở dữ liệu và mạng máy tính. Thứ tự các chương cũng là trình tự nội dung nên học. Chương 1 là nội dung căn bản, cần phải được học đầu tiên. Nếu có định hướng trở thành lập trình viên phát triển mặt trước, sinh viên cần đào sâu kiến thức ở các chương từ 2 đến 4, đồng thời nắm được các công cụ được giới thiệu ở Chương 5. Ngược lại, nếu có định hướng trở thành lập trình viên phát triển mặt sau, sinh viên cần đào sâu kiến thức ở các chương từ 6 đến 9, đồng thời nắm được các công cụ được giới thiệu ở Chương 10. Nếu có định hướng trở thành lập trình viên phát triển cả mặt trước và mặt sau (full-stack), sinh viên cần nắm vững kiến thức được trình bày trong tất cả các chương. Các công cụ được giới thiệu ở hai chương 5 và 10 đang là cập nhật ở thời điểm giáo trình được viết (năm 2017-2018) nhưng rất có thể sẽ trở nên lỗi thời và được thay thế bởi những công cụ khác trong vài năm tới. Do vậy, việc nắm vững các kiến thức nền tảng được trình bày ở các chương 2-4 (đối với phát triển mặt

trước), 6-9 (đối với phát triển mặt sau) là vô cùng quan trọng. Nắm vững những kiến thức và công nghệ nền tảng ở các chương 2-4, 6-9, cùng với việc sẵn sàng tiếp cận công cụ và kỹ thuật phát triển mới là yêu cầu bắt buộc đối với mọi lập trình viên phát triển ứng dụng web.

Trong toàn bộ giáo trình, sau mỗi phần trình bày kiến thức luôn có các chương trình, mã nguồn minh họa. Tất cả chương trình, mã nguồn này đều đã được tác giả lập trình, chạy thử trên ít nhất một vài trình duyệt. Sinh viên được khuyến cáo lặp lại việc lập trình, chạy thử chương trình, mã nguồn đó trong quá trình học nhằm làm sâu sắc hơn kiến thức tiếp thu được, đồng thời tự rèn luyện kỹ năng thực hành. Do hầu hết các mục trong giáo trình đều có chương trình, mã nguồn minh họa nên tất cả các chương trình, mã nguồn minh họa không được đánh số tường minh như các hình vẽ mà ngầm định được đánh số theo/trùng với chỉ số của đề mục. Ví dụ, chương trình minh họa trong Mục 3.2.1 được đánh số ngầm định là L.3.2.1 (hay Listing 3.2.1). Trường hợp có nhiều chương trình, mã nguồn minh họa trong cùng một mục, các chương trình, mã nguồn vẫn được đánh số ngầm định theo quy tắc ở trên, cộng thêm hậu tố cho biết thứ tự của chương trình, mã nguồn minh họa trong cùng mục. Ví dụ, Mục 4.4.2 có hai mã nguồn được đánh số ngầm định là L.4.4.2-1 và L.4.4.2-2. Sinh viên cũng có thể duyệt, tải về và chạy thử tất cả các chương trình, mã nguồn minh họa tại địa chỉ *http://uet.vnu.edu.vn/~thanhld/textbooks/webappdev/*.

Các tác giả cảm ơn các đồng nghiệp tại Trường Đại học Công nghệ, ĐHQGHN đã luôn khuyến khích các hoạt động nghiên cứu phát triển và giảng dạy về phát triển ứng dụng web.

Các tác giả bày tỏ sự cảm ơn đặc biệt đối với TS. Nguyễn Thanh Hùng (Trường Đại học Bách khoa Hà Nội), TS. Nguyễn Đình Hóa và TS. Nguyễn Trọng Khánh (Học viện Công nghệ Bưu chính Viễn thông), PGS.TS. Trương Anh Hoàng, PGS.TS. Nguyễn Đình Việt, TS. Hoàng Xuân Tùng, TS. Nguyễn Hoài Sơn, TS. Võ Đình Hiếu và ThS. Đào Minh Thư (Trường Đại học Công nghệ, ĐHQGHN) đã đọc bản thảo và cho ý kiến phản biện, góp ý chi tiết để các tác giả bổ sung, hoàn thiện, nâng cao chất lượng cuốn giáo trình này.

Vì giáo trình được xuất bản lần đầu nên khó tránh khỏi những hạn chế. Các tác giả mong nhận được những góp ý để lần xuất bản sau được hoàn chỉnh hơn. Các ý kiến góp ý, phê bình xin vui lòng gửi về địa chỉ *thanhld@vnu.edu.vn* hoặc *vietanh@vnu.edu.vn*.

Các tác giả trân trọng cảm ơn mọi ý kiến góp ý.

Hà Nội, ngày 05 tháng 7 năm 2018.

Lê Đình Thanh, Nguyễn Việt Anh

Chương 1
KIẾN TRÚC ỨNG DỤNG WEB

1.1. ỨNG DỤNG WEB CÙNG CÁC THÀNH PHẦN LIÊN QUAN

Nhằm đáp ứng nhu cầu sử dụng ngày càng lớn và đa dạng của người dùng, nhiều phần mềm ứng dụng (application software, viết tắt là application hay app) đã ra đời, trong đó có **ứng dụng web** (web application hay web app). Ứng dụng web là một dạng phần mềm ứng dụng rất phổ biến ngày nay. Google.com, Twitter.com, Facebook.com, Amazon.com, Github.com hay Wikipedia.org là một vài ví dụ trong rất nhiều ứng dụng web nổi tiếng, có hàng triệu người sử dụng trên khắp thế giới. Các tổ chức, doanh nghiệp, ở bất kể quy mô nào, hầu như đều sở hữu những dụng ứng dụng web riêng nhằm cung cấp tin tức, quảng bá sản phẩm, xử lý nghiệp vụ, giao dịch với khách hàng. Nhiều cá nhân cũng là chủ sở hữu của ít hay nhiều các ứng dụng web khác nhau.

Ứng dụng web có những đặc trưng khác biệt so với các dạng phần mềm ứng dụng còn lại[1]. *Thứ nhất*, giao diện hay tương tác với người dùng của ứng dụng web không được thực hiện trực tiếp mà gián tiếp thông qua các phần mềm trung gian là **trình duyệt web** (web browser), gọi tắt là trình duyệt, và **trình phục vụ web** (web server), gọi tắt là trình phục vụ. Người dùng tương tác với trình duyệt, trình duyệt giao tiếp với trình phục vụ, và trình phục vụ giao tiếp với ứng dụng web. Thông qua chuỗi tương tác và giao tiếp như vậy, người dùng có thể gửi yêu cầu đến ứng dụng web, và ở chiều ngược lại ứng dụng web có thể trả kết quả xử lý về cho người dùng. Trình duyệt chạy ngay trên thiết bị của người dùng trong khi trình phục vụ và ứng dụng web thường chạy trên máy tính ở xa trên Internet. Trình duyệt và người dùng tạo nên **bên khách** (client-side) hay **mặt trước** (frontend), trong khi trình phục vụ và ứng dụng web thuộc về **bên phục vụ** (server-side) hay **mặt sau** (backend). Giao tiếp giữa trình duyệt và trình phục vụ web được thực hiện theo giao thức **HTTP** (Hypertext Transfer Protocol) theo mô hình **khách-phục vụ** (client-server).

Thứ hai, kết quả xử lý hay đầu ra (output) của ứng dụng web, sản phẩm mà bên khách yêu cầu, còn gọi là **nội dung web** (web content), không phải là dữ liệu như những ứng dụng khác, mà chủ yếu là mã nguồn (source code) được thể hiện bằng các ngôn ngữ lập trình **HTML**, **JavaScript** và **CSS**. Trình duyệt web có nhiệm vụ **thông dịch** (interpret) mã nguồn nhận được và trình diễn kết quả cho người dùng. Nội dung web có chứa các **siêu liên kết** (hyperlink) và trên giao diện của trình duyệt, người dùng có thể chọn theo (follow) các siêu liên kết để chuyển từ nội dung này đến nội dung khác.

---

[1] Ứng dụng console, desktop, hay mobile.

Theo Collins English Dictionary năm 2012, ứng dụng web là *phần mềm ứng dụng được truy cập trên Internet*. Random House Dictionary năm 2017 đưa ra một định nghĩa chi tiết hơn về ứng dụng web. Theo đó, ứng dụng web là *phần mềm cung cấp chức năng tương tác và được truy cập thông qua trình duyệt web và URL*. Các định nghĩa này ẩn vai trò của trình phục vụ web, HTTP và đặc trưng của nội dung web. Do vậy, ứng dụng web nên được nhận diện theo các đặc trưng đã được trình bày ở trên.

Nội dung mà ứng dụng web cung cấp cho trình khách được tạo ra từ các **tài nguyên web** (web resource). Từ góc độ lưu trữ, ứng dụng web là tập hợp các tài nguyên web. Tài nguyên là bất kỳ thứ gì có thể tạo ra nội dung. Tài nguyên thường gặp nhất là tệp. Tài nguyên cũng có thể là một tiến trình với một socket và một dòng dữ liệu ra. Có thể hình dung tài nguyên là đơn vị sản xuất ra nội dung, và bên phục vụ có nhiều đơn vị như vậy. Mỗi lần gửi yêu cầu tới trình phục vụ, trình duyệt chỉ định rõ một tài nguyên duy nhất nó cần.

Trình duyệt ít khi yêu cầu tài nguyên không phải là HTML ngay từ đầu. Thay vào đó, trình duyệt sẽ bắt đầu với việc yêu cầu một tài liệu HTML, phân tích nội dung HTML nhận được rồi yêu cầu các tài nguyên được tham chiếu bởi HTML. Một tài liệu HTML cùng các tài nguyên được nó tham chiếu được gọi là **trang web** (web page). Một cách logic, ứng dụng web là tập các trang web có quan hệ mật thiết với nhau về chức năng, dữ liệu và người dùng. Cấu trúc tài liệu HTML sẽ được trình bày trong Chương 2.

Một trang web không chỉ chứa các siêu liên kết đến các trang trong cùng ứng dụng mà còn có thể có những siêu liên kết đến các trang thuộc ứng dụng khác. Đông đảo các trang web có mặt trên Internet cùng số lượng lớn các siêu liên kết giữa chúng tạo nên một mạng lưới, một không gian thông tin hết sức phong phú và rộng khắp gọi là **World Wide Web** (viết tắt là WWW hoặc the Web). Internet, trình phục vụ web, trình duyệt web và giao thức HTTP là những thành phần quan trọng tạo nên hạ tầng cho việc xây dựng không gian thông tin này, hay là hạ tầng cho việc triển khai và khai thác các ứng dụng web.

Cần phân biệt ứng dụng web với **website**. Website là thuật ngữ để chỉ một nhóm các trang web được kết nối với nhau và cùng mang thông tin về một chủ đề hay một nhóm các chủ đề liên quan gần gũi với nhau. Các trang web thuộc cùng website không nhất thiết phải thuộc cùng ứng dụng web.

1.2. TRÌNH KHÁCH WEB

Ngoài trình duyệt, các phần mềm khác như telnet, wget, crawler cũng yêu cầu và sử dụng nội dung do ứng dụng web tạo ra. Trình duyệt và các phần mềm có sử dụng nội dung web được gọi chung là **trình khách web** (web client). Các trình khách sử dụng nội dung web theo nhiều cách khác nhau. Nếu như trình duyệt

thông dịch mã nguồn nhận được và hiển thị kết quả, tương tác với người dùng thì crawler tìm kiếm nội dung trên mã nguồn, telnet và wget đơn giản chỉ kéo mã nguồn về máy tính của người dùng. Do tính phổ biến rõ rệt của trình duyệt, trong giáo trình này, nếu không có phát biểu rõ ràng, trình khách web được hiểu là trình duyệt web, hai thuật ngữ "trình duyệt (web)" và "trình khách (web)" có thể được sử dụng thay thế cho nhau. Các trình duyệt điển hình bao gồm Mozilla Firefox, Google Chrome, Microsoft Edge, Apple Safari.

1.3. TRÌNH PHỤC VỤ WEB

Trình phục vụ web còn được gọi với tên khác là **trình phục vụ HTTP** (HTTP server) vì giao thức truyền thông giữa trình phục vụ web và trình khách là HTTP. Một mặt, trình phục vụ có nhiệm vụ giao tiếp với trình khách để nhận các yêu cầu từ trình khách và gửi đáp ứng cho trình khách. Mặt khác, trình phục vụ web phải tương tác với ứng dụng web để chuyển tiếp yêu cầu đến ứng dụng và nhận kết quả xử lý từ ứng dụng. Cùng một lúc, trình phục vụ web phải giao tiếp với nhiều trình khách, đồng thời tương tác với nhiều ứng dụng web. Các trình phục vụ web phổ biến bao gồm Apache, IIS, và Nginx.

1.4. TRÌNH PHỤC VỤ ỨNG DỤNG

Cũng như các ứng dụng khác, ứng dụng web cần có môi trường để thực thi. Môi trường đó được cung cấp bởi **trình phục vụ ứng dụng** (application server). Trình phục vụ ứng dụng bao gồm các ngôn ngữ lập trình và các thư viện thực thi (runtime libraries). Ứng dụng web là mã chạy trên trình phục vụ ứng dụng và được viết bằng ngôn ngữ được trình phục vụ ứng dụng hỗ trợ. Ứng dụng web gọi đến các thư viện thời gian thực thi cùng các thành phần khác được trình phục vụ ứng dụng cung cấp.

Giao tiếp giữa trình phục vụ ứng dụng với trình phục vụ web có thể được thực hiện thông qua **CGI** (Common Gateway Interface). Ví dụ, ActivePerl và Apache được kết hợp theo cách này. Theo CGI, trình phục vụ ứng dụng đứng sau trình phục vụ web, chạy độc lập với trình phục vụ web như một ứng dụng bàn giao tiếp (console hay shell) được nối ống (pipe, trên Linux ký hiệu là |) vào/ra với trình phục vụ web. Các nối ống vào/ra giúp chuyển tiếp yêu cầu từ trình phục vụ web đến trình phục vụ ứng dụng cũng như chuyển kết quả từ trình phục vụ ứng dụng đến trình phục vụ web. Hạn chế của CGI là tiêu tốn tài nguyên và tốc độ xử lý chậm do việc khởi tạo các tiến trình phục vụ ứng dụng. Nhằm khắc phục những hạn chế này, các trình phục vụ web hiện nay đều cho phép tích hợp các trình phục vụ ứng dụng như những môđun và giao tiếp với chúng thông qua **API** (Application Programming Interface). Nói cách khác, trình phục vụ ứng dụng được nhúng trong trình phục vụ web và hai trình phục vụ hòa nhập vào nhau,

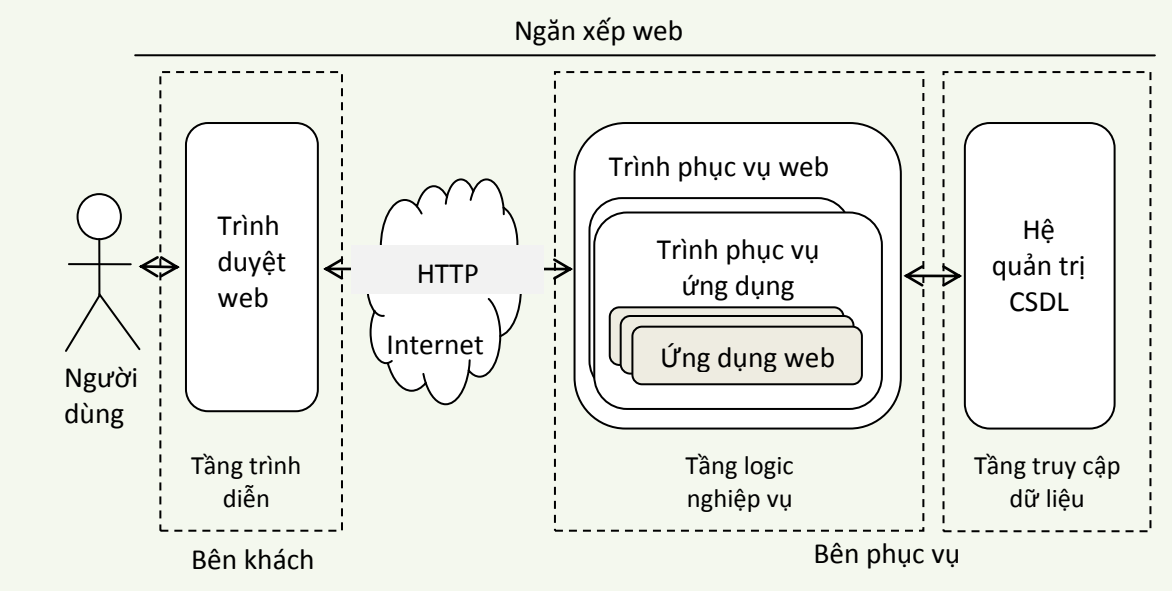
được gọi bằng một tên chung là trình phục vụ web. Ví dụ, Apache không chỉ có httpd (phục vụ web) mà còn có mod_php, mod_python, mod_perl, … để phục vụ ứng dụng.

1.5. HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Một thành phần nữa không thể thiếu trong các ứng dụng web ngày nay là cơ sở dữ liệu (CSDL) cho lưu trữ bền vững. Hầu hết các hệ quản trị CSDL phổ biến hiện nay như MariaDB/MySQL, PostgreSQL, MS SQL, Oracle, hay JavaDB đều có thể sử dụng cho ứng dụng web. Trình phục vụ ứng dụng cung cấp các giao diện (interface) cho phép ứng dụng web kết nối đến các CSDL khác nhau và thao tác dữ liệu. Hệ quản trị CSDL thường được cài đặt trên máy tính trong cùng mạng cục bộ (LAN) với máy tính chạy trình phục vụ web nhằm đảm bảo tốc độ truy xuất dữ liệu. Hệ quản trị CSDL cũng thuộc về bên phục vụ.

1.6. KHUNG NHÌN CHUNG

Đến lúc này, tất cả các thành phần liên quan ứng dụng web đều đã được giới thiệu. Nhằm cung cấp bức tranh toàn cảnh về ứng dụng web, một khung nhìn chung cho tất cả các thành phần được vẽ ra như Hình 1.1. Khung nhìn ấy phản ánh đầy đủ các thành phần cũng như tương tác giữa các thành phần.



Hình 1.1. Kiến trúc ứng dụng web.

Ứng dụng web được xây dựng theo **kiến trúc đa tầng** (multitier hoặc n-tier architecture), trong đó mỗi tầng thực hiện một số chức năng riêng, cung cấp dịch vụ cho tầng liền trên, và sử dụng dịch vụ được cung cấp bởi tầng liền dưới. Kiến trúc đa tầng được áp dụng cho ứng dụng web thường bao gồm ba tầng (three-

tier): trình diễn, logic nghiệp vụ và truy cập dữ liệu. **Tầng trình diễn** (presentation) bao gồm trình khách, có nhiệm vụ chính là trình diễn kết quả và tương tác với người dùng. **Tầng logic nghiệp vụ** (business logic) bao gồm trình phục vụ web và ứng dụng web, có nhiệm vụ giải quyết bài toán, nhận yêu cầu giải quyết từ tầng trình diễn và trả kết quả cho tầng trình diễn. **Tầng truy cập dữ liệu** (data access) thường bao gồm hệ quản trị CSDL, có nhiệm vụ lưu trữ và thao tác dữ liệu, nhận yêu cầu thao tác dữ liệu từ tầng logic nghiệp vụ và trả dữ liệu cho tầng logic nghiệp vụ.

Yêu cầu xử lý thường phát sinh từ tầng trình diễn và do người dùng đưa ra. Trong khi xử lý yêu cầu này, tầng trình diễn gọi đến tầng logic nghiệp vụ. Tiếp đó, tầng logic nghiệp vụ gọi đến tầng truy cập dữ liệu. Tầng truy cập dữ liệu xử lý xong thì tầng logic nghiệp vụ mới có dữ liệu và xử lý tiếp. Tầng logic nghiệp vụ xử lý xong thì tầng trình diễn mới có kết quả và trình diễn cho người dùng. Quá trình phát sinh và kết thúc của các yêu cầu xử lý trên các tầng, như vừa mô tả, có hình ảnh của một ngăn xếp (stack). Do vậy, kiến trúc đa tầng cho ứng dụng web còn được gọi là **ngăn xếp web** (web stack). Các tầng trình diễn, logic nghiệp vụ và truy cập dữ liệu là những phần tử trong ngăn xếp ấy.

1.7. ĐỊNH DANH ỨNG DỤNG WEB

Một trình phục vụ web có thể lưu trữ (**hosting**) nhiều ứng dụng web. Do vậy, các ứng dụng web cần được gán định danh để phân biệt ứng dụng này với ứng dụng khác. Một ứng dụng web cần có ít nhất một định danh, nhưng có thể có nhiều định danh. Ngược lại, một định danh chỉ được gán cho duy nhất một ứng dụng. Các trình phục vụ web ngày nay thường sử dụng kết hợp lược đồ (scheme, là http hoặc https), địa chỉ IP, số hiệu cổng (TCP port) và tên máy (hostname) chạy trình phục web để làm định danh cho ứng dụng web. Việc gán định danh cho ứng dụng web, theo đó, còn được gọi là **buộc** (binding) lược đồ, địa chỉ IP, số hiệu cổng và tên miền cho ứng dụng. Lý do sử dụng phương thức định danh này có liên quan đến giao thức HTTP. HTTP sử dụng TCP ở tầng giao vận, có thể qua tầng trung gian TLS (lược đồ https). Gói TCP được gửi từ trình khách đến trình phục vụ web có các trường thông tin về địa chỉ IP và số hiệu cổng đích của máy phục vụ. Tải (payload) của nó là yêu cầu HTTP có chứa thông tin về tên của máy phục vụ. Do vậy, căn cứ vào gói tin nhận được, trình phục vụ web có thể nhận biết được yêu cầu trong gói tin đó dành cho ứng dụng nào và chuyển yêu cầu đến đúng ứng dụng đó.

Có thể buộc nhiều tổ hợp bất kỳ của lược đồ, địa chỉ IP, số hiệu cổng, và tên máy cho một ứng dụng. Tuy nhiên, trong thực hành, ba cách thức thường được thực hiện là buộc theo IP (IP-based), theo cổng (port-based) hoặc theo tên (name-based). Buộc theo IP sử dụng địa chỉ IP làm định danh, không phân biệt số hiệu cổng và tên máy. Máy chạy trình phục vụ web phải có nhiều giao diện mạng mới

thực hiện được cách thức buộc này. Buộc theo cổng sử dụng số hiệu cổng làm định danh, không phân biệt địa chỉ IP và tên máy. Buộc theo tên sử dụng tên máy làm định danh. Buộc theo tên được ưa chuộng hơn do người dùng chỉ cần nhớ tên miền để truy cập ứng dụng.

Thiết lập định danh cho ứng dụng web bằng cách thức buộc như mô tả ở trên còn được gọi với tên khác là **lưu trữ ảo** (virtual hosting) vì với việc thay đổi địa chỉ IP, số hiệu cổng hay tên miền, người dùng nhìn thấy các ứng dụng web có vẻ như được phục vụ từ các máy hay các trình phục vụ khác nhau, nhưng thực tế chúng có thể được phục vụ bởi cùng một trình phục vụ.

1.8. ĐỊNH VỊ TÀI NGUYÊN WEB

Tiếp theo ứng dụng web, từng tài nguyên web cũng cần được định danh. Hơn nữa, địa chỉ và cách thức truy cập tài nguyên web cũng cần được xác định. **URL**[2] (Uniform Resource Locator hay Định vị tài nguyên đồng nhất) được sử dụng để xác định địa chỉ và cách thức truy cập tài nguyên, không chỉ web, trên Internet. Các tài nguyên được truy cập thông qua URL của chúng. URL có cấu trúc như sau:

scheme://host[:port][/path][?query-string][#bookmark]

Sáu thành phần có thể có trong URL lần lượt là lược đồ (scheme), địa chỉ máy (host), số hiệu cổng (port), đường dẫn (path), chuỗi truy vấn (query-string), và điểm đánh dấu (bookmark), trong đó các thành phần bắt buộc là lược đồ và địa chỉ máy. Lược đồ cho biết cách thức truy cập tài nguyên. Đối với tài nguyên web, hai lược đồ có thể áp dụng là http và https. Nếu lược đồ http được sử dụng, giao tiếp HTTP sẽ được thực hiện trực tiếp trên kết nối TCP. Nếu lược đồ **https** được sử dụng, giao tiếp HTTP được đào hầm (tunneled) trong giao thức bảo mật TLS (Transport Layer Security), TLS sử dụng kết nối TCP. Địa chỉ máy được xác định bằng tên miền hoặc địa chỉ IP của máy chạy trình phục vụ web. Số hiệu cổng là cổng đã được buộc cho ứng dụng. **Cổng chuẩn mặc định** cho ứng dụng web là 80 với lược đồ http và 443 với lược đồ https. Nếu cổng chuẩn mặc định được sử dụng, thành phần số hiệu cổng có thể vắng mặt. Trong trường hợp không sử dụng cổng chuẩn mặc định, số hiệu cổng là thành phần bắt buộc. Đường dẫn là xâu ký tự dùng để nhận diện tài nguyên bên trong ứng dụng. Nếu không có đường dẫn, **tài nguyên mặc định** (default document) của ứng dụng web sẽ được chỉ định. Đường dẫn bắt đầu bằng dấu chéo trái (/) và bao gồm các xâu con được phân cách bởi dấu chéo trái. Lưu ý, đường dẫn không nhất thiết phải là đường dẫn vật lý, tức đường dẫn của tệp tài nguyên được lưu trên máy phục vụ. Đường dẫn có thể là một bí danh (alias) và sẽ được trình phục vụ web ánh xạ đến một tài nguyên vật lý cụ thể. Chuỗi truy vấn được phân cách với các thành phần phía trước bằng dấu

[2] URL là một dạng của URI (Uniform Resource Identifier).

chấm hỏi (?), bao gồm các cặp *tham_số=giá_trị* được phân cách với nhau bởi ký tự '&'. Chuỗi truy vấn cung cấp dữ liệu của người dùng (user input) cho ứng dụng web. Tùy theo chuỗi truy vấn, một tài nguyên web có thể trả về những nội dung khác nhau. Cuối cùng, điểm đánh dấu, được phân cách với các thành phần phía trước bằng dấu thăng (#), là định danh của một phần tử trong trang web. Trình duyệt sẽ điều chỉnh vị trí thanh cuộn để người dùng có thể nhìn thấy phần tử có định danh là điểm đánh dấu.

URL chỉ được bao gồm các ký tự ASCII in được, tức có mã từ 0x20 đến 0x7e. Hơn nữa, một số ký tự, ví dụ '/' và '#', được dành riêng (reserved) vì chúng có nghĩa đặc biệt trong URL. Tuy nhiên, giá trị của các tham số được người dùng nhập vào URL có thể chứa các ký tự bất kỳ. Do vậy, mỗi ký tự không được phép phải được biểu diễn hay thay thế bằng một chuỗi các ký tự được phép. Trình duyệt sẽ tự động làm điều này trước khi gửi yêu cầu HTTP. Trình phục vụ sẽ thực hiện phép thay thế ngược lại để khôi phục giá trị của tham số. Việc biểu diễn một ký tự bất kỳ bằng một chuỗi các ký tự được phép trong URL được gọi là **biểu diễn mã URL** (URL encoding). Theo cách biểu diễn này, mỗi ký tự thuộc bảng mã ASCII được biểu diễn bằng mã hexa của nó với hai chữ số và dấu phần trăm (%) ở phía trước. Ví dụ, các ký tự '/' và '#', có mã tương ứng là 0x23 và 0x2f, được biểu diễn mã URL là %23 và %2f, tương ứng. Dấu phần trăm được viết đầu các mã hexa trong các biểu diễn. Do vậy, biểu diễn mã URL còn được gọi là **biểu diễn mã phần trăm** (percent-encoding). Để URL hợp lệ, tất cả các ký tự dành riêng (theo RFC 3986, bao gồm *'();:@&=+$,/?#[]) phải được biểu diễn mã phần trăm. Các ký tự không dành riêng được khuyến cáo giữ nguyên. Các ký tự ngoài bảng mã ASCII, nếu xuất hiện trong URL, được chuyển đổi thành dãy bytes, mỗi byte được biểu diễn bằng một mã phần trăm. Ký tự '%', nếu nằm trong URL với nghĩa gốc của nó, cũng phải được biểu diễn bằng mã phần trăm, tức %25.

URL có chuỗi truy vấn hoặc đường dẫn vật lý đến tài nguyên được gọi là **URL phi ngữ nghĩa** (non-semantic URL). Nhược điểm của URL phi ngữ nghĩa là mang đậm chi tiết cài đặt, ít có ngữ nghĩa và không thân thiện với người dùng cũng như không thân thiện với các máy tìm kiếm web (search engine). Tính khả dụng và khả truy cập của nó, do vậy, bị hạn chế. URL phi ngữ nghĩa ngày càng ít được sử dụng. Ngược lại, **URL ngữ nghĩa** (semantic URL) ngày càng được sử dụng rộng rãi. URL ngữ nghĩa che đi các chi tiết cài đặt, mặt khác cung cấp cấu trúc khái niệm (conceptual structure) của ứng dụng, thân thiện với cả người dùng và máy tìm kiếm. Với URL ngữ nghĩa, chuỗi truy vấn không còn được sử dụng. Thay vào đó, giá trị của các tham số được nhập vào đường dẫn, làm cho URL trở nên ngắn gọn và sáng sủa (clean). Sự tương phản giữa URL ngữ nghĩa và phi ngữ nghĩa có thể được quan sát qua một vài ví dụ sau.

URL phi ngữ nghĩa	URL ngữ nghĩa
http://example.com/index.php?page=marketing	http://example.com/marketing
http://example.com/services.py?cat=legal	http://example.com/services/legal

Các kỹ thuật **viết lại** (rewrite) và **định tuyến URL** (URL routing) được sử dụng để tạo URL ngữ nghĩa. Các kỹ thuật này sẽ được trình bày trong Chương 9.

1.9. HTTP

HTTP là một giao thức truyền thông quan trọng được sử dụng để truy cập World Wide Web và được tất cả các ứng dụng web ngày nay sử dụng. Giao thức này sử dụng mô hình dựa trên thông điệp (message-based model) trong đó trình khách web gửi một thông điệp yêu cầu HTTP (HTTP request) và trình phục vụ web trả lại một thông điệp đáp ứng HTTP (HTTP response). HTTP hoàn toàn phi kết nối (connectionless) mặc dù nó sử dụng giao thức hướng kết nối, có trạng thái (connection-oriented, stateful) TCP ở tầng giao vận. Phiên bản HTTP được sử dụng phổ biến hiện nay là HTTP/1.1. Trước HTTP/1.1 có các phiên bản HTTP/1.0 và HTTP/0.9. Sau HTTP/1.1 có HTTP/2 mới được công bố năm 2015.

1.9.1. Yêu cầu HTTP

Mỗi yêu cầu HTTP bao gồm một dòng yêu cầu (request line), một hoặc nhiều tiêu đề (headers), mỗi tiêu đề nằm trên một dòng. Hết các tiêu đề là một dòng trắng, và có thể một thân (body). Ví dụ một yêu cầu HTTP như sau:

```
GET /auth/488/Details.php?uid=129 HTTP/1.1
Accept: application/x-ms-application, image/jpeg, */*
Referer: http://uet.vnu.edu.vn/auth/488/Home.php
Accept-Language: en
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Accept-Encoding: gzip, deflate
Host: uet.vnu.edu.vn
Connection: Keep-Alive
Cookie: SessionId=5BFH0C71F3FD4945635CDB6682E549176
```

Dòng yêu cầu phải là dòng đầu tiên của yêu cầu HTTP và bao gồm ba thành phần được phân cách bởi dấu cách lần lượt là động từ (verb) chỉ phương thức HTTP, URL của tài nguyên được yêu cầu, phiên bản HTTP được sử dụng. Các phương thức HTTP cùng các tiêu đề sẽ được trình bày trong các mục nhỏ phía sau. Lưu ý, dòng trắng sau các tiêu đề là dòng bắt buộc.

1.9.2. Đáp ứng HTTP

Đáp ứng HTTP bao gồm một dòng trạng thái (status line), một hoặc nhiều tiêu đề, mỗi tiêu đề nằm trên một dòng, theo sau là một dòng trắng, và một thân. Ví dụ một đáp ứng HTTP như sau:

```
HTTP/1.1 200 OK
Date: Tue, 19 Apr 2011 09:23:32 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc
X-AspNet-Version: 2.0.50727
Cache-Control: no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1067

<!DOCTYPE html><html><head><title>Your details</title>
...
```


Dòng trạng thái phải là dòng đầu tiên của đáp ứng HTTP và bao gồm ba thành phần được phân cách bởi dấu trắng lần lượt là phiên bản HTTP được sử dụng, mã trạng thái, và diễn giải mã trạng thái. Các mã trạng thái sẽ được trình bày trong các mục nhỏ phía sau. Thân của đáp ứng HTTP chứa nội dung được trả về cho trình khách. Lưu ý, dòng trắng sau các tiêu đề cũng là dòng bắt buộc.

1.9.3. Phương thức HTTP

Phương thức HTTP (HTTP method), còn được gọi là động từ (verb), được ghi trong yêu cầu HTTP nhằm xác định cách thức xử lý yêu cầu này tại trình phục vụ. GET và POST là hai phương thức được sử dụng thường xuyên hơn cả. Phương thức GET được thiết kế để trình khách nhận tài nguyên từ trình phục vụ. Khi người sử dụng nhập một URL trên thanh địa chỉ của trình duyệt và nhấn Enter, hay kích chuột vào một liên kết trong trang web, trình duyệt sẽ gửi một yêu cầu HTTP theo phương thức GET cho trình phục vụ. Yêu cầu HTTP theo phương thức GET không có phần thân. Tuy nhiên, có thể sử dụng chuỗi truy vấn trong URL để gửi tham số đến tài nguyên được yêu cầu. Tài nguyên được yêu cầu có thể căn cứ vào giá trị của các tham số để trả về những nội dung khác nhau cho trình khách. Vì độ dài của URL bị giới hạn, độ dài của chuỗi truy vấn cũng bị giới hạn theo, tức số lượng tham số và dung lượng của giá trị tham số bị giới hạn.

Ngược lại với GET, phương thức POST được thiết kế để trình khách gửi dữ liệu cho trình phục vụ và đề nghị trình phục vụ chấp nhận dữ liệu đó. Với phương thức này, các tham số và giá trị của chúng được đặt trong thân của yêu cầu HTTP. Tất nhiên, vẫn có thể đặt chuỗi truy vấn, tức tham số và giá trị, vào URL. Phương thức POST được sử dụng khi cần upload tệp hay gửi dữ liệu của người dùng với dung lượng lớn. Một khuyến cáo được đưa ra là sử dụng phương thức GET với những yêu cầu không làm thay đổi trạng thái bên phục vụ (ví dụ, tra cứu điểm thi

của một sinh viên) và sử dụng phương thức POST nếu yêu cầu làm thay đổi trạng thái bên phục vụ (ví dụ, chỉnh sửa điểm thi của một sinh viên).

Ngoài hai phương thức GET và POST, giao thức HTTP hỗ trợ các phương thức khác cho những mục đích cụ thể. Phương thức HEAD có chức năng giống như GET, ngoại trừ đáp ứng HTTP trả về chỉ có các tiêu đề mà không có thân. HEAD có thể được sử dụng để kiểm tra liệu một tài nguyên có tồn tại hay không. Cũng có thể sử dụng HEAD để biết thời điểm cập nhật cuối của tài nguyên, từ đó biết được tài nguyên có được cập nhật so với bản lưu đệm (cached) ở trình duyệt hay không. Nếu có, trình duyệt mới cần gửi yêu cầu GET để lấy bản cập nhật về. Ngược lại, trình duyệt có thể dùng ngay bản lưu trong đệm. Phương thức TRACE được thiết kế cho mục đích chẩn đoán, lần vết. Với TRACE, thân của đáp ứng HTTP chứa nguyên vẹn nội dung yêu cầu HTTP mà nó đã nhận. Điều đó có thể được sử dụng để phát hiện liệu yêu cầu HTTP có bị thay đổi trên đường truyền đến bên phục vụ hay không. Phương thức OPTIONS được sử dụng để hỏi các phương thức HTTP được trình phục vụ hỗ trợ cho một tài nguyên cụ thể. Trình phục vụ sẽ trả về đáp ứng HTTP có tiêu đề Allow liệt kê các phương thức mà nó hỗ trợ. Phương thức PUT được sử dụng để upload tệp lên trình phục vụ. Nếu PUT được hỗ trợ, trình khách có thể upload tệp bất kỳ, có thể là tệp kịch bản chương trình. Do vậy, vì lý do an ninh, PUT nên hạn chế được sử dụng.

1.9.4. Tiêu đề HTTP

HTTP hỗ trợ một số lượng lớn các tiêu đề. Một số tiêu đề có thể được sử dụng cho cả yêu cầu và đáp ứng HTTP. Các tiêu đề còn lại hoặc chỉ áp dụng cho yêu cầu hoặc chỉ áp dụng cho đáp ứng.

Một số tiêu đề chung:

Content-Encoding – cho biết nội dung trong thân của thông điệp được biểu diễn mã như thế nào.

Content-Type – cho biết kiểu MINE của nội dung chứa trong thân của thông điệp (xem thêm Mục 1.10).

Một số tiêu đề cho yêu cầu:

Cookie – đệ trình cookies.

Host – xác định tên máy trong URL đang được yêu cầu.

Referer – cho biết URL đã chuyển đến yêu cầu hiện tại.

User-Agent – cung cấp thông tin về trình duyệt.

Một số tiêu đề cho đáp ứng:

Access-Control-Allow-Origin – cho biết có thể truy cập tài nguyên từ miền khác hay không.

Expires – cho biết trình duyệt có thể lưu đệm thân đáp ứng đến khi nào.

Server – cung cấp thông tin về trình phục vụ web.

Set-Cookie – yêu cầu trình duyệt lưu cookies và gửi lại cookies trong các yêu cầu sau.

1.9.5. Mã trạng thái

Mỗi đáp ứng HTTP phải mang một mã trạng thái trong dòng đầu tiên, cho biết kết quả xử lý yêu cầu. Các mã trạng thái rơi vào năm nhóm như sau:

1xx – Cung cấp thông tin.

2xx – Yêu cầu thành công.

3xx – Trình khách được chuyển hướng sang một tài nguyên khác.

4xx – Yêu cầu có lỗi.

5xx – Trình phục vụ có lỗi và không thể đáp ứng yêu cầu.

Có nhiều mã trạng thái khác nhau, rất nhiều trong số đó chỉ được dùng trong những tình huống cụ thể. Dưới đây là những mã trạng thái hay gặp nhất:

200 - *OK* – Yêu cầu thành công và thân của đáp ứng chứa kết quả xử lý.

301 - *Moved Permanently* – Chuyển hướng vĩnh viễn trình duyệt đến URL mới được chỉ định trong tiêu đề *Location* của đáp ứng.

400 - *Bad Request* – Yêu cầu không hợp lệ, chẳng hạn không đúng cấu trúc.

401 - *Unauthorized* – Trình phục vụ yêu cầu xác thực HTTP trước.

403 - *Forbidden* – Không được truy cập đến tài nguyên được yêu cầu.

404 - *Not Found* – Không tìm thấy tài nguyên.

500 - *Internal Server Error* – Trình phục vụ gặp lỗi khi xử lý yêu cầu.

503 - *Service Unavailable* – Trình phục vụ vẫn làm việc nhưng ứng dụng không sẵn sàng.

1.9.6. Kết nối liên tục và cơ chế dẫn ống

HTTP sử dụng giao thức giao vận TCP để truyền tải yêu cầu và đáp ứng của nó. Nếu mỗi kết nối TCP chỉ được sử dụng để gửi và nhận một yêu cầu và đáp ứng tương ứng thì thời gian đáp ứng cho nhiều yêu cầu sẽ rất lớn do phải nhiều lần mở và đóng các kết nối TCP. Giải pháp cho vấn đề này là sử dụng lại một kết nối TCP đã mở để gửi và nhận nhiều yêu cầu/đáp ứng tiếp theo trước khi đóng kết nối TCP lại. Việc sử dụng một kết nối TCP để gửi, nhận nhiều yêu cầu và đáp ứng HTTP như vậy được gọi là **kết nối HTTP liên tục** (HTTP persistent connection). Tất cả trình duyệt hiện đại sử dụng kết nối liên tục. Một vấn đề trong

sử dụng kết nối liên tục là các yêu cầu HTTP sẽ được gửi đồng bộ hay không đồng bộ trên kết nối đó. Gửi đồng bộ có nghĩa là trình duyệt sẽ phải đợi cho đến khi nhận được đáp ứng của yêu cầu trước thì mới được gửi yêu cầu tiếp theo. Ngược lại, gửi không đồng bộ nghĩa là trình duyệt có thể gửi liên tiếp nhiều yêu cầu mà không cần phải đợi đáp ứng của các yêu cầu trước đó. Trong truyền thông, gửi không đồng bộ còn được gọi là **dẫn ống** (pipelining). Rõ ràng, dẫn ống làm tăng hiệu suất truyền thông, do vậy tăng hiệu năng sử dụng. Tuy nhiên, ngược lại với kết nối liên tục, cơ chế dẫn ống hầu như không được các trình duyệt ngày nay hỗ trợ hoặc bị các trình duyệt vô hiệu hóa theo mặc định. Lý do đằng sau thực tế này là các nhà phát triển e ngại cơ chế dẫn ống chưa được hỗ trợ đầy đủ bởi các trình phục vụ cũng như các phần mềm trung gian khác.

1.10. KIỂU MINE

Ngoài HTML, JavaScript và CSS, ứng dụng web có thể cung cấp cho trình khách nội dung khác dưới dạng tài nguyên được tham chiếu. Ví dụ, ứng dụng web có thể trả về cho trình khách một bản PDF, một ảnh JPEG, một dòng âm thanh MP3 hay một đoạn mã Java bytecode. Căn cứ vào **kiểu nội dung** (content type), còn được gọi là **kiểu phương tiện** (media type) hay **kiểu MINE** (MINE type)[3], trình duyệt sẽ lựa chọn ứng dụng xử lý phù hợp. Các ứng dụng xử lý có thể được tích hợp sẵn trong trình duyệt hoặc được gọi và chạy bên ngoài trình duyệt. Ví dụ, ứng dụng xem ảnh được tích hợp trong hầu hết các trình duyệt nhưng ứng dụng phát audio/video chỉ được tích hợp trong một số trình duyệt hiện đại; nếu nhận được một dòng audio/video mà trình duyệt không có ứng dụng phát phù hợp đã được tích hợp sẵn, trình duyệt buộc phải tải dòng dữ liệu đa phương tiện xuống rồi gọi một trình phát nào đó có trên máy tính thực hiện phát tệp audio/video đã được tải về. Một số kiểu MINE phổ biến và quan trọng đối với ứng dụng web là[4] *text/html, text/javascript, text/css, image/gif, image/jpeg, image/png, audio/mp3, video/mp4, text/xml, application/json, text/plain*. Thông thường, các ứng dụng web phục vụ được hầu hết các kiểu nội dung. Tuy nhiên, trình phục vụ web có thể từ chối phục vụ một kiểu nội dung cụ thể nào đó. Đồng thời, trình phục vụ web cũng cung cấp khả năng cấu hình danh sách các kiểu nội dung mà nó phục vụ.

1.11. WEB TĨNH VÀ WEB ĐỘNG

Một trang web được gọi là tĩnh (static) nếu nội dung của nó không thay đổi, được trả về như nhau đối với tất cả yêu cầu. Nói cách khác, mọi trình khách ghé thăm trang web tĩnh, ở bất kỳ thời điểm nào, đều nhận được cùng một nội dung.

[3] MINE (Multipurpose Internet Mail Extensions) là một chuẩn Internet.

[4] Danh sách đầy đủ các kiểu MINE được IANA công bố tại https://www.iana.org/assignments/media-types/media-types.xhtml

Thông thường, trang web tĩnh được tạo thành từ một nguồn lưu trữ bền vững (tệp, CSDL) lưu sẵn mã nguồn HTML, JavaScript và CSS. Khi nhận được yêu cầu, ứng dụng web chỉ cần đọc nội dung từ nguồn lưu trữ bền vững và đưa nội dung vào thân của đáp ứng HTTP. Ngược lại, một trang web được gọi là động (dynamic) nếu nội dung của nó thay đổi theo từng yêu cầu. Với thời điểm yêu cầu khác nhau, tiêu đề hay tham số khác nhau trong các yêu cầu, một trang web động có thể trả về nội dung khác nhau. Khác với trang web tĩnh, trang web động tạo ra nội dung (mã nguồn HTML, JavaScript và CSS) khi có yêu cầu (on-demand). Trang web động thường được tạo thành bằng các kịch bản ở các ngôn ngữ lập trình đa năng như PHP, Perl, Python, Java, C#. Khi được yêu cầu, kịch bản được thông dịch và chạy để tạo ra nội dung HTML, JavaScript và CSS. Một ứng dụng web có thể bao gồm cả các trang web tĩnh và các trang web động.

Lưu ý, tính chất động hay tĩnh được dùng để chỉ nội dung được thay đổi hay không thay đổi giữa các lần đáp ứng khác nhau. Tính chất động hay tĩnh cũng có thể được hiểu là nội dung HTML, JavaScript và CSS được tạo ra theo yêu cầu (on-demand) hay được chuẩn bị trước (prepared).

Ưu điểm của trang web tĩnh là đơn giản và hiệu năng cao. Do nội dung có sẵn mà không cần xử lý để có nội dung, ứng dụng web có thể đáp ứng rất nhanh các yêu cầu gọi đến trang web tĩnh. Trình phục vụ web có thể nén sẵn các trang web tĩnh để tiết kiệm dung lượng, thời gian truyền dữ liệu. Các trình khách cũng dễ dàng lưu đệm các trang web tĩnh. Tất cả những điều đó góp phần làm tăng hiệu năng của trang web tĩnh. Tuy nhiên, hạn chế của trang web tĩnh là khó cung cấp các chức năng động và linh hoạt. Ngược lại, trang web động cung cấp khả năng tùy biến cao trong cung cấp nội dung và chức năng nhưng hạn chế về mặt hiệu năng, phức tạp trong tổ chức. Các ứng dụng web ngày nay biết tận dụng lợi thế của cả trang web tĩnh và trang web động bằng cách sử dụng các bộ tạo web tĩnh (Static site generator) để tạo ra các trang web tĩnh từ nội dung động.

1.12. WEB PROXY

Ngoài các thành phần đã được giới thiệu ở các mục trước, một thành phần khác có thể xuất hiện trong các hệ thống web là web proxy. Proxy là thành phần trung gian nằm giữa trình khách và trình phục vụ và được sử dụng với nhiều mục đích khác nhau.

Mục đích thứ nhất của việc sử dụng web proxy, xuất phát từ người sử dụng, là tăng tốc độ truy cập và giảm chi phí duyệt web. Với mục đích này, proxy được triển khai ngay trong mạng LAN của cơ quan, nơi có nhiều người cùng truy cập web. Proxy sẽ lưu đệm mọi trang web mà nó biết, bất kể trang web đó được phục vụ từ trình phục vụ nào. Mọi yêu cầu duyệt web xuất phát từ các máy trong cơ quan được chuyển đến proxy. Nếu proxy tìm thấy trang web được yêu cầu có trong đệm của nó, proxy sẽ trả về cho trình khách bản đệm được tìm thấy. Ngược

lại, proxy sẽ chuyển tiếp yêu cầu đến trình phục vụ, nhận đáp ứng, chuyển đáp ứng cho trình khách, và lưu đệm trang web vừa được kéo về. Các yêu cầu tiếp sau đến cùng trang web sẽ không cần phải chuyển đến trình phục vụ nữa. Như vậy, một mặt thời gian đáp ứng được tăng lên một cách rõ rệt, mặt khác băng thông sử dụng Internet được tiết kiệm vì nhiều yêu cầu không cần được gửi ra Internet. Các proxy được sử dụng theo mục đích vừa mô tả được gọi là **forward proxy** vì nó thực hiện chuyển tiếp các yêu cầu khi cần thiết.

Mục đích thứ hai của việc sử dụng web proxy, xuất phát từ nhà cung cấp, là cân bằng tải, tăng tốc độ phục vụ và tăng độ an toàn khi phục vụ web. Với mục đích này, proxy được triển khai trước với các trình phục vụ web, thay mặt cho các trình phục vụ web tiếp nhận và đáp ứng mọi yêu cầu từ các trình khách. Proxy thực hiện cân bằng tải và lưu đệm các trang được phục vụ từ các trình phục vụ mà proxy thay mặt. Các trình phục vụ không được truy cập trực tiếp bởi trình khách nên giảm được các rủi ro tấn công vào lỗ hổng an ninh của trình phục vụ hay hệ điều hành chạy trình phục vụ. Tất nhiên, proxy không thể làm giảm rủi ro tấn công vào lỗ hổng an ninh của chính ứng dụng web. Các proxy được sử dụng theo mục đích thứ hai được gọi là **reverse proxy**.

1.13. LỊCH SỬ PHÁT TRIỂN WEB

Đầu thập niên 90, Tim Berners-Lee công tác tại phòng thí nghiệm khoa học CERN đã đề xuất hệ thống quản lý thông tin có thể chia sẻ tài nguyên thông qua mạng máy tính, được gọi là World Wide Web, với mục tiêu ban đầu để chia sẻ tài liệu và thông tin của nhóm nghiên cứu. Trong những ngày đầu, các trang web chỉ là tĩnh, được viết bằng HTML.

Với sự phát triển của nhu cầu sử dụng, xây dựng các trang web động có sự tương tác trở thành một yêu cầu. Giải pháp phổ biến là sử dụng ngôn ngữ lập trình để xây dựng ứng dụng web sinh ra nội dung web động. Giao tiếp giữa ứng dụng web với trình phục vụ web lúc đầu chỉ là CGI.

Năm 1995, Netscape giới thiệu JavaScript. Cùng năm, Sun phát triển công nghệ servlet cho phép thực thi chương trình Java bên phục vụ thay vì applet chạy bên khách. Servlet cung cấp thư viện hoàn chỉnh để thao tác với giao thức HTTP.

Năm 1996, Macromedia giới thiệu công nghệ Flash, có thể được thêm vào trình duyệt như một plugin để nhúng hình ảnh vào trang web.

Cuối những năm 90, hàng loạt công nghệ phát triển ứng dụng web ra đời như JSP, ASP và PHP. Các công nghệ lập trình này hỗ trợ việc phát triển ứng dụng thuận tiện hơn.

Những năm gần đây chứng kiến sự ra đời và áp dụng nhiều công nghệ phát triển ứng dụng web như ASP.NET, Ruby, Python, Go; đồng thời chứng kiến sự bùng nổ của các khung phát triển (frameworks) ứng dụng web như Laravel,

Symfony, Rails, ASP.NET MVC, Django, NodeJS, AngularJS, Ember, jQuery, Bootstrap; cũng như chứng kiến các mô hình phát triển mới như MVC (Model-View-Controller), SPA (Single Page Application), Thick Client – Thin Server.

Bài tập

1. Hãy xác định các thành phần trong URL sau:
   http://vnu.edu.vn:81/daotao/monhoc.php?mamon=15&hanhdong=xem#mota
2. Sử dụng trình duyệt Firefox để truy cập trang web tại địa chỉ *http://vnu.edu.vn*. Đồng thời, sử dụng công cụ Network Monitor được tích hợp sẵn trong Firefox để xem nội dung các yêu cầu và đáp ứng HTTP đã được trình duyệt gửi và nhận. Trang web có bao nhiêu tài nguyên? Tên tệp và kiểu MINE của ba tài nguyên được trả về trong ba đáp ứng HTTP đầu tiên lần lượt là gì?
3. Cài đặt trình phục vụ web Nginx và sử dụng cấu hình mặc định của nó. Thư mục nào là thư mục gốc của ứng dụng web? Những tài nguyên nào là tài nguyên mặc định của ứng dụng? Sử dụng trình duyệt để duyệt trang web mặc định được tạo cùng cài đặt Nginx.

Đọc thêm

1. R. Fielding et al., "RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1", The Internet Society, 1999.
2. David Gourley and Brian Totty, "HTTP: The Definative Guide", O'Reilly Media, 2002.
3. Stephen Ludin and Javier Garza, "Learning HTTP/2: A Practical Guide for Beginners", O'Reilly Media, 2017.

Chương 2

XÂY DỰNG TRANG WEB BẰNG HTML

2.1. MÃ NGUỒN TRANG WEB, ĐỐI TƯỢNG TÀI LIỆU

Từ góc độ lập trình, trang web là kịch bản với mã nguồn được viết bằng các ngôn ngữ lập trình HTML, CSS và JavaScript[5]. Bên phục vụ có nhiệm vụ tạo ra và cung cấp mã nguồn cho trình khách, trong khi trình khách có nhiệm vụ *thông dịch* mã nguồn nhận được. Về bản chất, trình duyệt web là trình thông dịch (interpreter). Chức năng của nó là thông dịch mã nguồn trang web và trình diễn kết quả trên giao diện người dùng.

Một câu hỏi được đặt ra hết sức tự nhiên là tại sao lại phải sử dụng tới ba ngôn ngữ lập trình (HTML, CSS, JavaScript), mà không phải là một, để tạo mã nguồn trang web? Câu hỏi này sẽ được giải đáp chi tiết trong các phần tiếp theo. Tuy nhiên, câu trả lời ngắn nhưng bao quát là "*Trang web là tập các đối tượng/phần tử tài liệu (document object/element); HTML được sử dụng để khai báo ra chúng, CSS được sử dụng để định kiểu trình diễn chúng trên giao diện, và JavaScript được sử dụng để quản lý chúng*". Sử dụng HTML chỉ có thể khai báo tĩnh, hay khởi tạo các đối tượng tài liệu lúc trang vừa được tải về, trong khi sử dụng JavaScript có thể quản lý và khai báo động các đối tượng tài liệu bất cứ khi nào. Mặc dù HTML có thể được sử dụng để định kiểu trình diễn cho các đối tượng tài liệu, với nguyên lý "*tách biệt nội dung với trình diễn*" trong kỹ nghệ phần mềm, các lập trình viên web ngày nay ưa chuộng và được khuyến cáo sử dụng CSS cho mục đích này.

2.2. THẺ

HTML (Hypertext Markup Language) là ngôn ngữ đánh dấu. Tuy nhiên, vượt xa mục đích thông thường của ngôn ngữ đánh dấu là cung cấp các *thẻ* (tag) cho phép định dạng văn bản, HTML được sử dụng để *khai báo* các đối tượng tài liệu. Các thẻ HTML thực sự là những câu lệnh khai báo. Mỗi đối tượng tài liệu được khai báo bằng một thẻ HTML. Thẻ có vai trò như *hàm tạo* (constructor) trong các ngôn ngữ lập trình hướng đối tượng. Khi gặp thẻ, đối tượng tài liệu sẽ được tạo. Thẻ cũng cho phép gán giá trị ban đầu cho các thuộc tính của đối tượng. Cú pháp khai báo đối tượng tài liệu sử dụng thẻ HTML như sau:

<tenThe thuoctinh1='giatri1' thuoctinh2="giatri2" ... />

hoặc

<tenThe thuoctinh1=giatri1 … > </tenThe>

[5] Có thể dùng VBScript, JScript, … thay cho JavaScript, nhưng các ngôn ngữ này không phổ biến.

Thẻ bắt đầu với dấu nhỏ hơn (<) và kết thúc bởi dấu lớn hơn (>). Về cú pháp, thẻ được chia thành thẻ đơn và thẻ đôi. Thẻ đôi là cặp thẻ bao gồm thẻ mở và thẻ đóng. Thẻ đóng có cùng tên với thẻ mở và có thêm ký tự chéo trái (/) ngay sau dấu nhỏ hơn. Thẻ đơn là thẻ chỉ có thẻ mở mà không có thẻ đóng. Để có sự nhất quán trong cú pháp, một vài phiên bản HTML (ví dụ XHTML) yêu cầu ký tự chéo trái được đặt ở cuối thẻ đơn, trước dấu lớn hơn, với ý nghĩa thẻ đơn vừa là thẻ mở vừa là thẻ đóng. Thông thường, tên thẻ là một từ, hoặc viết tắt của một từ trong tiếng Anh, ví dụ *form, img, div*. Tên thẻ không phân biệt chữ hoa và thường, ví dụ *FORM* và *form* là như nhau. Thuộc tính được khai báo trong thẻ mở. Thứ tự liệt kê các thuộc tính không quan trọng, tức không ảnh hưởng đến kết quả của câu lệnh khai báo. Giá trị thuộc tính được đặt giữa hai dấu nháy đơn (') hoặc hai dấu nháy kép ("). Có thể không cần sử dụng các dấu nháy đơn và nháy kép nếu giá trị là số hoặc xâu ký tự không có dấu trắng. Nếu một thuộc tính không được liệt kê trong thẻ, giá trị của nó sẽ được đặt mặc định. Theo nghĩa này, mỗi thẻ HTML chính là một hàm tạo với nhiều giá trị mặc định.

Một đối tượng tài liệu được khai báo bằng thẻ đôi có thể chứa bên trong nó các đối tượng khác. Đối tượng bên ngoài được gọi là *đối tượng cha* (parent element). Đối tượng chứa bên trong được gọi là *đối tượng con* (child element). Ví dụ sau đây khai báo ba đối tượng lần lượt là <*label*>, <*textarea*> và <*input*> nằm trong đối tượng <*form*>:

```
1. <form>
2.   <label>Bài viết: </label>
3.   <textarea name="newpost"></textarea>
4.   <input type="submit" value="Gửi"/>
5. </form>
```

Một đối tượng con lại có thể chứa các đối tượng con của nó. Cứ như vậy, các đối tượng tài liệu được khai báo bằng HTML có quan hệ với nhau theo cấu trúc *cây* (tree).

Ngoài các thẻ hay các lệnh khai báo đối tượng tài liệu, HTML không cung cấp thêm các lệnh nào khác. Sử dụng HTML chỉ có thể tạo ra các đối tượng tài liệu chứ không thể quản lý (thay đổi thuộc tính, triệu gọi phương thức, loại bỏ, …) được chúng. Hơn nữa, hầu hết các đối tượng tài liệu đều thuộc các lớp dựng sẵn (built-in), và được gọi là đối tượng chuẩn (standard element). Khai báo các đối tượng chuẩn bằng HTML sẽ được trình bày trong các mục nhỏ dưới đây. Định nghĩa lớp đối tượng tài liệu mới, hay lớp đối tượng tùy biến (custom element), là nội dung dành cho tự học và không được trình bày trong giáo trình này.

2.3. CHÚ THÍCH

Chú thích trong HTML được viết như sau. Nội dung chú thích được đặt giữa <!-- và -->.

```
1. <!-- Đây là chú thích
2. trong HTML -->
```

2.4. CẤU TRÚC TÀI LIỆU HTML

Toàn bộ tài liệu HTML được thể hiện bằng một đối tượng *<html>*. Đối tượng này có đúng hai đối tượng con theo thứ tự lần lượt là *<head>* (phần đầu) và *<body>* (phần thân). Đối tượng *<head>* chứa các đối tượng con mang siêu dữ liệu (meta data). Đối tượng *<body>* chứa các đối tượng con mang nội dung của trang web. Chỉ các đối tượng trong *<body>* mới được hiển thị trên giao diện của trình duyệt.

```
1. <!DOCTYPE html>
2. <html>
3.   <head>
4.     <!-- Khai báo các đối tượng phần đầu. -->
5.   </head>
6.   <body>
7.     <!-- Khai báo các đối tượng phần thân. -->
8.   </body>
9. </html>
```

2.5. DOCTYPE

<!DOCTYPE> không phải là một thẻ, mà là một chỉ dẫn (directive). Chỉ dẫn này, nếu có, phải được đặt ngay đầu tài liệu HTML, trước tất cả các thẻ/lệnh khai báo đối tượng tài liệu. Chỉ dẫn này cho biết phiên bản HTML được sử dụng trong tài liệu HTML, giúp trình duyệt hiển thị đúng trang web. Với phiên bản HTML nhỏ hơn 5, chỉ dẫn <!DOCTYPE> có tham chiếu đến DTD (Document Type Definition). Với HTML 5, <!DOCTYPE> không cần tham chiếu đến DTD. Ví dụ khai báo <!DOCTYPE> cho HTML 5 và HTML 4.01 Strict lần lượt như sau:

```
1. <!DOCTYPE HTML>
```

```
1. <!DOCTYPE HTML PUBLIC "-
   //W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

2.6. TIÊU ĐỀ TRANG

Đối tượng *<title>* được sử dụng để khai báo tiêu đề cho cả tài liệu HTML. Mỗi tài liệu HTML chỉ có duy nhất một đối tượng *<title>* và *<title>* phải nằm trong *<head>*. Khai báo đối tượng *<title>* như sau:

```
1. <title>Nội dung tiêu đề<title>
```

Tiêu đề sẽ được hiển thị trên thanh tiêu đề hoặc tab của trình duyệt khi tài liệu HTML được tải.

2.7. THÔNG TIN VỀ TRANG

Các đối tượng *<meta>* cung cấp thông tin, còn gọi là siêu dữ liệu, về tài liệu HTML. Chúng không được hiển thị trên giao diện người dùng nhưng có tác dụng hướng dẫn trình duyệt thông dịch và trình diễn kết quả. Ví dụ, đối tượng *<meta> charset* với câu lệnh khai báo là *<meta charset="utf-8"/>* hướng dẫn trình duyệt phân tích và hiển thị văn bản theo bảng mã UTF-8. Meta cũng được dùng bởi các chương trình tìm kiếm hay dịch vụ web khác.

Ngoại trừ *<meta> charset* có ý nghĩa như đã được mô tả ở trên, các đối tượng *<meta>* phân biệt với nhau bằng thuộc tính *name* hoặc (nhưng không đồng thời) thuộc tính *http-equiv*. Giá trị của các thuộc tính này cho biết tác dụng hay ý nghĩa của *<meta>*. Nội dung của *<meta>* được được xác định bằng thuộc tính *content*. Meta theo *name* cho biết thông tin về nội dung của trang trong khi meta theo *http-equiv* cho biết các tiêu đề HTTP. Những *<meta>* thường xuyên được sử dụng được liệt kê và mô tả trong ví dụ dưới đây.


```
1. <!-- Tên của ứng dụng -->
2. <meta name="application-name" content="Tuyển sinh"/>
3. <!-- Tác giả -->
4. <meta name="author" content="Nguyễn Văn A"/>
5. <!-- Mô tả trang -->
6. <meta name="description" content="Thông tin tuyển sinh"/>
7. <!-- Các từ khóa của trang -->
8. <meta name="keywords" content="tuyển sinh, chỉ tiêu, điểm sàn"/>
9. <!-- Kích thước và tỉ lệ phóng to/thu nhỏ vùng hiển thị -->
10. <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
11. <!-- Kiểu nội dung và bảng mã được sử dụng trong trang -->
12. <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
```

Các đối tượng *<meta>* phải nằm trong đối tượng *<head>*. Có thể có nhiều đối tượng *<meta>* trong cùng một tài liệu HTML.

2.8. LIÊN KẾT TÀI NGUYÊN

Đối tượng *<link>* được sử dụng để liên kết tài liệu hiện thời (tài liệu chứa link) với các tài nguyên khác, gọi là tài nguyên được tham chiếu. Quan hệ giữa tài liệu hiện thời và tài nguyên được tham chiếu được xác định bằng thuộc tính *rel* (relationship). Tùy thuộc vào quan hệ, tài nguyên được tham chiếu có thể trở thành một bộ phận của tài liệu hiện thời hoặc tách biệt với tài liệu hiện thời. Các quan hệ giữa tài liệu hiện thời và tài nguyên được tham chiếu như: *alternate* (Tài nguyên được tham chiếu là biểu diễn khác của tài liệu hiện thời), *up* (Tài nguyên được tham chiếu cung cấp ngữ cảnh cho tài liệu hiện thời), *stylesheet* (Tài nguyên được tham chiếu là CSS và sẽ được nhập vào trang hiện thời), … URL của tài nguyên được tham chiếu được xác định bằng thuộc tính *href*. Ngoài ra, thuộc tính *type* cho biết kiểu MIME của tài nguyên được tham chiếu và thuộc tính *hreflang*

cho biết ngôn ngữ của văn bản trong tài nguyên được tham chiếu. Dưới đây là một vài ví dụ sử dụng các đối tượng <*link*>.

```
1. <link rel=stylesheet type="text/css" href="default.css"/>
2. <link rel=alternate type="text/html" href="/en-
   us" hreflang=en title="English"/>
```

Các đối tượng <*link*> phải nằm trong đối tượng <*head*>. Có thể có nhiều đối tượng <*link*> trong cùng một tài liệu HTML.

2.9. KỊCH BẢN

Đối tượng <*script*> được sử dụng để bao hàm hay định nghĩa kịch bản phía trình khách. Nó có thể trực tiếp chứa các câu lệnh kịch bản hoặc tham chiếu đến một tệp kịch bản. Thuộc tính *src* (source) được sử dụng để tham chiếu đến tệp kịch bản. Nếu <*script*> tham chiếu đến tệp kịch bản, nội dung của nó phải để rỗng. Có thể sử dụng thuộc tính *type* để cho biết kiểu MINE (giá trị mặc định là "text/javascript") của tệp kịch bản và thuộc tính *charset* để cho biết bảng mã được sử dụng trong tệp kịch bản. Nếu *charset* không được xác định, tệp kịch bản được hiểu là sử dụng cùng bảng mã với tài liệu HTML. Dưới đây là một vài ví dụ khai báo đối tượng <*script*>.


```
1. <script type="text/javascript" src="form.js"></script>
2. <script>
3.   var n = document.getElementById("note");
4.   n.innerHTML = "Một bài viết đã được xóa.";
5. </script>
```

Ngoài đặt trong <*head*>, <*script*> cũng có thể nằm trong <*body*>. Có thể có nhiều đối tượng <*script*> trong cùng một tài liệu HTML. Sử dụng JavaScript sẽ được trình bày chi tiết trong Chương 4.

2.10. KIỂU TRÌNH DIỄN

Đối tượng <*style*> được sử dụng để định nghĩa các bảng định dạng CSS. Dưới đây là ví dụ khai báo đối tượng <*style*>.

```
1. <style type="text/css">
2.   body {background-color:#eee;}
3.   .container {position:relative;}
4.   #note {color:red;}
5. </style>
```

Các đối tượng <*style*> được đặt trong đối tượng <*head*>. Có thể có nhiều đối tượng <*style*> trong cùng một tài liệu HTML. Sử dụng CSS sẽ được trình bày chi tiết trong Chương 3.

2.11. NỘI DUNG VĂN BẢN

Các đối tượng thuộc phần thân mang nội dung của trang web và chiếm một tỉ lệ lớn trong các tài liệu HTML. Mục này và các mục tiếp theo trình bày các đối tượng nội dung thuộc phần thân của tài liệu HTML.

2.11.1. Đầu mục

Đầu mục (heading) văn bản được khai báo bằng các đối tượng *<h1>*-*<h6>*. *<h1>* là đầu mục lớn nhất. *<h6>* là đầu mục nhỏ nhất. Ví dụ sau là khai báo và hiển thị năm đầu mục từ lớn đến nhỏ.

```
1. <h1>Nội dung web</h1>
2. <h2>Khai báo tài liệu HTML</h2>
3. <h3>Nội dung phần thân</h3>
4. <h4>Khai báo và trình bày văn bản</h4>
5. <h5>Đầu mục</h5>
```

2.11.2. Văn bản thường

Văn bản (text) thường là đối tượng tài liệu đặc biệt vì không có và không cần bất kỳ thẻ nào để khai báo ra nó. Dãy các ký tự nằm giữa hai thẻ liên tiếp tạo thành đối tượng văn bản thường. Ví dụ, đoạn mã sau khai báo một đối tượng đầu mục *<h1>* và một đối tượng văn bản bên trong *<h1>* có nột dung là "Nội dung web".

```
1. <h1>Nội dung web</h1>
```

Ví dụ khác, đoạn mã ngắn sau đây khai báo năm đối tượng, trong đó có ba đối tượng văn bản: một đối tượng *<h1>*, một đối tượng *<h2>*, một đối tượng văn bản nằm trong *<h1>* có nội dung là "Nội dung web", một đối tượng văn bản nằm trong *<h2>* có nội dung là "Khai báo tài liệu HTML", và một đối tượng văn bản nằm giữa *<h1>* và *<h2>* có nội dung là " " (ba dấu cách).

```
1. <h1>Nội dung web</h1>   <h2>Khai báo tài liệu HTML</h2>
```

Khi trình diễn các đối tượng văn bản, theo mặc định, các từ (dãy ký tự liên tiếp không bao gồm dấu cách và dấu xuống dòng) của văn bản được hiển thị từ trái sang phải, từ trên xuống dưới, các từ trên cùng dòng cách nhau một dấu trắng.

2.11.3. Trích dẫn

Đối tượng *<cite>* được sử dụng để chỉ tên công trình được trích dẫn. Nếu cần thể hiện nội dung được trích dẫn, các đối tượng *<q>* và *<blockquote>* có thể được sử dụng. *<q>* được dùng nếu nội dung trích dẫn ngắn. Ngược lại, *<blockquote>* được dùng để trích dẫn cả đoạn có nội dung dài. Có thể chỉ định nguồn trích dẫn bằng việc gán URL cho thuộc tính *cite* của *<q>* và *<blockquote>*. Ví dụ khai báo một số trích dẫn như sau.

```
1. Tác phẩm <cite>The Scream</cite> được Edward Munch vẽ năm 1893.
2. Phương châm của jQuery là <q cite="https://jquery.com">Write less, do more
   </q>.
3. <blockquote cite="https://tools.ietf.org/html/rfc7230"> The Hypertext
   Transfer Protocol (HTTP) is a stateless application-level
   protocol for distributed, collaborative, hypertext information systems.
   </blockquote>
```

2.11.4. Đoạn văn

Đối tượng <*p*> (paragraph) được sử dụng để tạo các đoạn văn. Theo mặc định, <*p*> được hiển thị theo khối, với độ rộng bằng độ rộng của vùng hiển thị. Dòng đầu của văn bản có trong <*p*> được giãn dòng rộng hơn, nhưng không có lùi dòng, so với các dòng còn lại. Ví sau là khai báo đối tượng <*p*>.

```
1. <p>Từ lâu, cây tre đã trở thành một trong những biểu tượng cực kỳ đẹp đẽ
   về sức sống và phẩm cách con người Việt Nam chúng ta. Nhà thơ Nguyễn Duy
   khẳng định điều này bằng những hình ảnh giàu sức khái quát và bằng cách
   nói dí dỏm, hợp với sự tiếp nhận của cả các bạn đọc nhỏ tuổi.</p>
```


2.11.5. Bài viết

Đối tượng <*article*> được sử dụng để định nghĩa một nội dung độc lập, nghĩa là nội dung có thể đứng một mình, không cần một ngữ cảnh khác, mà vẫn giữ nguyên đầy đủ ý nghĩa. Bài viết trên diễn dàn, bản tin, bài báo là những ví dụ về nội dung độc lập. Ví dụ khai báo đối tượng <*article*> như sau.

```
1. <article>
2.   <h1>Thông báo tuyển sinh đại học năm 2017</h1>
3.   <p>Năm 2017, Trường Đại học Công nghệ tuyển sinh 5000 chỉ tiêu đại học
    và hơn 1000 chỉ tiêu sau đại học.</p>
4. <article>
```

2.11.6. Phân đoạn tài liệu

Một bài viết có thể được chia thành nhiều phân đoạn. Đối tượng <*section*> được sử dụng để thực hiện việc này. Nội dung trong <*section*> có quan hệ về mặt ngữ nghĩa với nhau. Ví dụ, chương, bài, mục trong bài viết là các <*section*>.

```
1. <section>
2.   <h3>Phân đoạn tài liệu</h3>
3.   <p> Một tài liệu HTML có thể được chia thành nhiều phân đoạn.</p>
4. </section>
```

2.11.7. Ngắt chủ đề

Đối tượng <*hr*> (horizontal rule) được sử dụng để ngắt chủ đề. Nếu trang web có nhiều nội dung với các chủ đề khác nhau, giữa hai nội dung thuộc hai chủ đề khác nhau là điểm thích hợp để đặt <*hr*>. Theo mặc định, các trình duyệt sẽ trình diễn <*hr*> như một đường kẻ ngang. Ví dụ sử dụng <*hr*> như sau.

```
1. <h1>HTML</h1>
2. <p>HTML dùng để khai báo các đối tượng tài liệu.....</p>
3. <hr>
```

```
4. <h1>CSS</h1>
5. <p>CSS định kiểu trình diễn các đối tượng tài liệu.....</p>
```

2.11.8. Tham chiếu ký tự

Mỗi ký tự trong HTML có thể tự biểu diễn nó hoặc được biểu diễn bằng một dãy các ký tự khác. Dãy các ký tự biểu diễn một ký tự được gọi là *tham chiếu ký tự* (character reference). Tham chiếu ký tự thường được sử dụng để biểu diễn các ký tự dành riêng (ví dụ <) hoặc ký tự nằm ngoài bảng mã ASCII (ví dụ β).

Có hai cách tham chiếu ký tự là theo tên hoặc theo số. Tham chiếu ký tự theo tên có dạng &*name*; trong đó *name* là tên hoặc viết tắt tên của ký tự. Ví dụ, các ký tự < (less-than) và β (beta) được biểu diễn là &*lt*; và &*beta*; tương ứng. HTML định nghĩa 252 tham chiếu ký tự theo tên[6]. Nhiều ký tự không có tham chiếu theo tên.

Tham chiếu ký tự theo số có dạng &#*dddd*; hoặc &#x*hhhh*; trong đó *dddd* và *hhhh* là mã của ký tự viết dưới hệ thập phân và hexa, tương ứng. Ví dụ, ký tự < được biểu diễn là < hoặc <, ký tự β được biểu diễn là β hoặc β. Khác với tham chiếu theo tên, tham chiếu theo số có cho mọi ký tự.

2.11.9. Sử dụng bảng mã

Để các ký tự được hiển thị đúng trên giao diện của trình duyệt, tài liệu HTML cần cho trình duyệt biết bảng mã (còn gọi là tập ký tự hay charset) nào được sử dụng cho trang web. Có thể sử dụng một trong hai <*meta*> sau đây để chỉ định bảng mã được dùng:

```
1. <meta charset="BẢNG_MÃ">
```

hoặc

```
1. <meta http-equiv="Content-Type" content="text/html; charset=BẢNG_MÃ">
```

Bảng mã được khuyến cáo sử dụng cho các trang web là UTF-8 vì với bảng mã này có thể thể hiện trang web ở bất kỳ ngôn ngữ nào.

2.12. SIÊU LIÊN KẾT, ĐIỂM ĐÁNH DẤU

Đối tượng <*a*> (anchor) định nghĩa siêu liên kết, được sử dụng để liên kết hay tham chiếu từ tài liệu hiện thời đến tài nguyên khác. Thuộc tính quan trọng nhất của nó là *href* chỉ URL của tài nguyên được tham chiếu. Đối tượng <*a*> cũng có các thuộc tính *rel*, *type* và *hreflang* tương tự <*link*>. Một thuộc tính nữa của <*a*> hay được sử dụng là *target*. Nó cho biết nơi mở tài nguyên được tham chiếu. Các giá trị của *target* như: *_blank* (mở tài nguyên được tham chiếu ở một cửa sổ hoặc tab mới), *_self* (mặc định, mở tài nguyên được tham chiếu trong cùng khung hiện thời),

[6] https://www.w3.org/TR/html4/sgml/entities.html

_parent (mở tài nguyên được tham chiếu ở khung cha) … Dưới đây là một vài ví dụ khai báo đối tượng *<a>*.

```
1. <a href="http://vnu.edu.vn" rel="external" target="_blank">Website Đại
   học Quốc gia Hà Nội</a>
2. <a href="#tochuc">Chuyển đến mục Cơ cấu tổ chức</a>
3. <a href="/tuyensinh.htm" target="_blank">Trang tuyển sinh</a>
4. <a href="/download/bieumau01.pdf" type="application/pdf">Tải biểu mẫu</a>
5. <a href="/en-us/" rel=alternate hreflang=en>English version</a>
```

Đối tượng *<a>* cũng có thể tham chiếu đến điểm đánh dấu (bookmark). Điểm đánh dấu được sử dụng cho phép người dùng chuyển đến một phần cụ thể trong trang web. Với những trang web dài, các điểm đánh dấu đặc biệt hữu ích. Để sử dụng điểm đánh dấu, đầu tiên tạo điểm đánh dấu bằng việc chỉ định định danh cho đối tượng được đánh dấu, ví dụ

```
1. <h2 id="publication">Công bố khoa học</h2>
```

Tiếp đó, tạo liên kết đến điểm đánh dấu, ví dụ


```
1. <a href="#publication">Xem công bố khoa học</a>
```

Khi người dùng kích chuột vào liên kết đến điểm đánh dấu, trình duyệt sẽ di chuyển thanh cuộn đến vị trí của điểm đánh dấu. Cũng có thể chuyển đến điểm đánh dấu trong một trang mới ngay sau khi trang mới được tải xong như ví dụ sau.

```
1. <a href="page2.htm#project">Đề tài khoa học</a>
```

2.13. DANH SÁCH, BẢNG BIỂU

2.13.1. Danh sách có thứ tự

Đối tượng *<ol>* (ordered list) được sử dụng để tạo danh sách có thứ tự. Nó là đối tượng chứa (container), chứa các đối tượng *<li>* (list item), hay các mục trong danh sách. Mặc định, các mục sẽ được đánh thứ tự tăng dần theo số nguyên bắt đầu từ 1. Có thể chỉ định đánh thứ tự các mục theo số nguyên/chữ cái in hoa/chữ cái in thường/số La Mã in hoa/số La Mã in thường bằng cách đặt giá trị thuộc tính *type* của *<ol>* là 1/A/a/I/i, tương ứng. Có thể sử dụng thứ tự giảm dần bằng cách đặt thuộc tính *reversed* của *<ol>*. Ngoài ra, có thể sử dụng thuộc tính *start* của *<ol>* để chỉ định giá trị bắt đầu (gán cho mục đầu tiên) của danh sách. Giá trị bắt đầu được chỉ định là số nguyên. Nó sẽ tự động được chuyển đổi sang các biểu diễn khác phù hợp với giá trị của *type*. Ví dụ sau là khai báo hai danh sách có các mục như nhau nhưng được đánh thứ tự khác nhau. Danh sách thứ nhất được đánh thứ tự tăng dần theo số nguyên bắt đầu từ 100. Danh sách thứ hai được đánh thứ tự giảm dần theo số La Mã in thường bắt đầu từ *c* (100 trong hệ thập phân).

```
1. <ol start=100>
2.   <li>Coffee</li>
3.   <li>Tea</li>
```

```
4.   <li>Milk</li>
5. </ol>
6. <ol type=i start=100 reversed=reversed>
7.      <li>Coffee</li>
8.    <li>Tea</li>
9. </ol>
```

2.13.2. Danh sách không có thứ tự

Đối tượng *<ul>* (unordered list) được sử dụng để tạo danh sách không có thứ tự. Nó là đối tượng chứa (container), chứa các đối tượng *<li>*, hay các mục trong danh sách. Ví dụ sau là khai báo một danh sách không có thứ tự.

```
1. <ul>
2.   <li>Coffee</li>
3.   <li>Tea</li>
4.   <li>Milk</li>
5. </ul>
```

2.13.3. Danh sách mô tả

Đối tượng *<dl>* (description list) được sử dụng để tạo danh sách các thuật ngữ cùng với mô tả của thuật ngữ. Nó là đối tượng chứa, chứa các đối tượng *<dt>* hay là thuật ngữ và *<dd>* hay là mô tả cho thuật ngữ. Ví dụ sau là khai báo một danh sách mô tả.

```
1. <dl>
2.   <dt>WWW</dt>
3.     <dd>World Wide Web</dd>
4.   <dt>HTTP</dt>
5.     <dd>Hypertext Markup Language</dd>
6. </dl>
```

2.13.4. Bảng biểu

Để trình bày dữ liệu theo dạng bảng trên web, đối tượng *<table>* được sử dụng. Bản thân *<table>* là đối tượng chứa. Dữ liệu trong bảng được chứa trong các đối tượng *<th>* (table header) và *<td>* (table data), trong đó các đối tượng *<th>* và *<td>* được nhóm theo hàng bằng đối tượng *<tr>* (table row). Ở dạng đơn giản nhất, *<table>* chỉ chứa các *<tr>* và *<td>*. Có thể thêm các ô tiêu đề, tức *<th>*, vào trong bảng. Theo trình bày thông thường, các ô tiêu đề thường xuất hiện ở các dòng đầu hoặc cột đầu bảng. Lưu ý, các hàng không nhất thiết phải có số ô (*<td>* và *<th>*) như nhau, hàng này có thể có nhiều ô hơn hàng khác. Có thể nhóm gộp các hàng trong bảng thành ba phần là đầu bảng, chân bảng và thân bảng. Các đối tượng *<thead>*, *<tfoot>*, *<tbody>* được sử dụng cho mục đích này. Nếu có mặt đồng thời, chúng phải xuất hiện theo thứ tự liệt kê trên. Trình duyệt có thể sử dụng các đối tượng này để cuộn nội dung thân bảng độc lập với đầu và chân bảng. Hơn nữa, khi in bảng lớn trải nhiều trên nhiều trang, các đối tượng này cho phép đầu và chân bảng được in lặp lại trên tất cả các trang. Ví dụ sau khai báo bảng có cả ba phần là đầu, chân và thân bảng.

```
1. <table>
2.   <thead>
3.     <tr>
4.       <th>Tháng</th>
5.       <th>Tiền lãi</th>
6.     </tr>
7.   </thead>
8.   <tfoot>
9.     <tr>
10.      <td>Tổng</td>
11.      <td>$180</td>
12.    </tr>
13.  </tfoot>
14.  <tbody>
15.    <tr>
16.      <td>Giêng</td>
17.      <td>$100</td>
18.    </tr>
19.    <tr>
20.      <td>Hai</td>
21.      <td>$80</td>
22.    </tr>
23.  </tbody>
24. </table>
```

Cũng có thể nhóm gộp các cột của bảng bằng đối tượng *<colgroup>*. Nhóm gộp các cột thường được sử dụng để áp dụng kiểu trình diễn cho tất cả các ô trong một hoặc nhiều cột thay vì phải chỉ định cho từng ô. Đối tượng *<colgroup>* có thuộc tính *span* được sử dụng để chỉ định số lượng cột được gộp nhóm. Nếu có mặt, *<colgroup>* phải xuất hiện trước các đối tượng *<thead>*, *<tbody>*, *<tfoot>* và *<tr>*. Dĩ nhiên, có thể khai báo nhiều đối tượng *<colgroup>* trong một đối tượng *<table>*. Nếu cần chỉ định những cột cụ thể trong nhóm cột, đối tượng *<col>* có thể được khai báo trong *<colgroup>* để đạt mục đích này. Đối tượng *<col>* cũng có thuộc tính *span* cho phép một *<col>* đại diện cho nhiều hơn một cột. Ví dụ sau sử dụng hai đối tượng *<colgroup>*, đối tượng thứ nhất nhóm gộp hai cột đầu trong khi đối tượng *<colgroup>* thứ hai sử dụng các *<col>* để gộp ba cột còn lại của bảng.

```
1. <table>
2.   <caption>Danh sách nhân viên</caption>
3.   <colgroup span="2" style="background:red;"></colgroup>
4.   <colgroup>
5.     <col span="2" style="background:green;">
6.     <col style="background:blue;">
7.   </colgroup>
8.   <tr>
9.     <th>TT</th>
10.    <th>Họ</th>
11.    <th>Tên</th>
12.    <th>Điện thoại</th>
13.    <th>Email</th>
14.  </tr>
15.  <tr>
16.    <td>1</td>
17.    <td>Hoàng</td>
18.    <td>Vinh</td>
```

```
19.     <td>0987 555 666</td>
20.     <td>vinh@example.com</td>
21.   </tr>
22. </table>
```

Mặt khác, có thể thêm chú thích bảng sử dụng đối tượng *<caption>*. Chú thích bảng, nếu có mặt, phải trước tất cả các đối tượng khác trong bảng.

Cuối cùng, mỗi ô (cả *<td>* và *<th>*) có thể trải ra trên nhiều hàng hoặc nhiều cột. Các thuộc tính *rowspan* và *colspan* được sử dụng cho việc trải ô. Khi một ô được trải trên nhiều cột, các ô phía sau nó trong cùng hàng sẽ được đẩy sang bên phải khi hiển thị. Chính vì vậy, một số ô cuối hàng sẽ được hiển thị trên những cột mới bên phải của bảng. Để hiển thị của bảng không bị xô lệch, cần điều chỉnh nội dung các ô trên hàng bị xô lệch, đồng thời bỏ các ô ở cuối hàng và được hiển thị trên cột mới. Ví dụ bảng sau có một ô được trải trên hai cột ở hàng thứ nhất, và hàng thứ nhất có ít hơn một ô so với các hàng còn lại.

```
1. <table>
2.   <tr>
3.     <th colspan="2">Họ và Tên</th>
4.     <th>Điện thoại</th>
5.   </tr>
6.   <tr>
7.     <td>Hoàng</td>
8.     <td>Vinh</td>
9.     <td>0987 555 666</td>
10.  </tr>
11.  <tr>
12.    <td>Trần</td>
13.    <td>Bách</td>
14.    <td>0912 333 444</td>
15.  </tr>
16. </table>
```

Tương tự, nếu một ô được trải trên nhiều hàng, các ô ở các hàng có ô bị chiếm và sau ô bị chiếm sẽ được đẩy sang bên phải một ô khi hiển thị. Để hiển thị của bảng không bị xô lệch, cần điều chỉnh nội dung các ô trên hàng bị xô lệch, đồng thời bỏ các ô ở cuối hàng và được hiển thị trên cột mới.

2.14. NỘI DUNG NHÚNG

2.14.1. Đối tượng nhúng

Đối tượng *<object>* được sử dụng để nhúng một nội dung bên ngoài vào tài liệu HTML. Nó biểu diễn một nội dung bên ngoài nói chung, có thể là hình ảnh, âm thanh, phim, Java applets, ActiveX, PDF, Flash, …, kể cả một tài liệu HTML khác. Các thuộc tính *data* và *type* của *<object>* cho biết URL và kiểu MINE của tài nguyên được nhúng. Có thể truyền các tham số cho đối tượng nhúng bằng cách sử dụng các đối tượng *<param>* bên trong *<object>*. Ví dụ nhúng hình ảnh, tài liệu PDF và Flash vào trang web như sau.

```
1. <object data="sun.jpg" type="image/jpeg"></object>
2. <object data="mar.pdf" type="application/pdf"></object>
3. <object data="movie.swf" type="application/x-shockwave-flash">
4.   <param name="theme" value="ocean">
5. </object>
```

Mặc dù có thể nhúng hầu như bất kỳ nội dung bên ngoài nào vào tài liệu HTML, sử dụng đối tượng *<object>* không thực sự dễ hiểu. Chính vì vậy, với những kiểu nội dung cụ thể, các đối tượng chuyên biệt được dùng như được trình bày trong các mục tiếp theo.

2.14.2. Hình ảnh

Hình ảnh là một trong những nội dung thường xuyên được nhúng vào các trang web. Đối tượng *<img>* (image) được sử dụng để khai báo hình ảnh. Thuộc tính *src* (source) của nó cho biết URL của ảnh. Bản thân ảnh là tài nguyên độc lập với tài liệu HTML. *<img>* có chức năng tham chiếu đến ảnh và giữ không gian hiển thị cho ảnh. Ảnh có thể được phục vụ từ cùng hoặc khác máy chủ, cùng hoặc khác ứng dụng với tài liệu HTML. Trong triển khai thực tế, đặc biệt với các ứng dụng có lượng truy cập lớn, để đảm bảo tốc độ truy cập, các ảnh thường được phục vụ từ máy chủ khác. Ví dụ, ảnh sau đây được hiển thị trên trang báo điện tử Dân trí (*http://dantri.com.vn*) được phục vụ tại *https://dantricdn.com*.


```
1. <img src="https://dantricdn.com/zoom/94_79/2017/dsc8279-
   1485247438534.jpg">
```

Tuy nhiên, với những ứng dụng có lượng truy cập nhỏ và vừa phải, các ảnh có thể được phục vụ bởi cùng ứng dụng phục vụ tài liệu HTML. Ví dụ, ảnh sau được phục vụ tại website của Trường Đại học Công nghệ, ĐHQGHN (*http://uet.vnu.edu.vn*).

```
1. <img src= "/coltech/sites/uet-happy-2017.jpg">
```

Đối tượng *<img>* chỉ tham chiếu đến một ảnh duy nhất. Ảnh này có thể được co dãn khi hiển thị cho phù hợp với thiết bị. Tuy nhiên, chất lượng hiển thị ảnh có thể bị ảnh hưởng. Để hiển thị ảnh tốt hơn, đối tượng *<picture>* được khuyến khích sử dụng. *<picture>* thiết lập tham chiếu đến nhiều ảnh và cho phép trình duyệt lựa chọn ảnh phù hợp nhất. Ví dụ khai báo một đối tượng *<picture>* như sau.

```
1. <picture>
2.   <source media="(min-width:650px)" srcset="large_fls.jpg">
3.   <source media="(min-width:465px)" srcset="small_fls.jpg">
4.   <img src="default_fls.jpg">
5.   <figcaption>Hình 1. Phối cảnh tòa nhà HH9.</figcaption>
6. </picture>
```

Tham chiếu đến ảnh được xác định bởi các đối tượng *<source>* cùng một đối tượng *<img>* chứa trong *<picture>*. Đối tượng *<source>* có các thuộc tính *srcset*, *type*, *media* và *sizes*. *srcset* và *type* cho biết URL và kiểu MINE của ảnh. *media* và *sizes* chứa truy vấn thiết bị và mô tả độ rộng, tương ứng. Trình duyệt sẽ sử dụng đối

tượng *source* đầu tiên có giá trị của các thuộc tính phù hợp và bỏ qua các đối tượng còn lại. *<img>* là đối tượng bắt buộc và phải nằm cuối danh sách. Trong trường hợp không có đối tượng *<source>* phù hợp, đối tượng *<img>* sẽ được sử dụng. Truy vấn thiết bị hay còn gọi là CSS có điều kiện được trình bày trong Chương 3, Mục 3.10. Ngoài ra, có thể thêm chú thích cho hình ảnh bằng đối tượng *<figcaption>*.

2.14.3. Âm thanh, phim

Đối tượng *<audio>*/*<video>* được sử dụng để nhúng âm thanh/phim, tương ứng. Trình phát âm thanh/phim sẽ được gọi khi trình duyệt gặp đối tượng *<audio>*/*<video>*. Ví dụ khai báo một đối tượng *<audio>* như sau.

```
1. <audio controls>
2.   <source src="rose.ogg" type="audio/ogg">
3.   <source src="rose.mp3" type="audio/mpeg">
4.   Your browser does not support the audio tag.
5. </audio>
```


Các đối tượng *<audio>*/*<video>* sử dụng đối tượng *<source>* để tham chiếu đến các tài nguyên âm thanh/phim. Đối tượng *<source>* có thuộc tính *src* chứa URL của tài nguyên âm thanh/phim và *type* cho biết định dạng âm thanh/phim. Đối tượng *source* đầu tiên có định dạng âm thanh/phim được trình duyệt hỗ trợ sẽ được sử dụng. Nếu trình duyệt không hỗ trợ đối tượng *<audio>*/*<video>*, văn bản được khai báo trong đối tượng *<audio>*/*<video>* sẽ được hiển thị. Đối tượng *<audio>*/*<video>* có các thuộc tính sau đây để điều khiển trình phát âm thanh/phim: *autoplay* (tự phát âm thanh/phim khi đã sẵn sàng), *controls* (cho sử dụng các nút điều khiển), *loop* (phát lặp lại), *muted* (tạm dừng phát). Các thuộc tính này nhận giá trị boolean, xuất hiện nghĩa là true, vắng mặt nghĩa là false. Ngoài ra, đối tượng *<video>* sử dụng các thuộc tính *width*, *height* để xác định độ rộng và chiều cao vùng hiển thị phim, sử dụng thuộc tính *poster* để tham chiếu đến ảnh nền (ảnh được hiển thị khi phim không/chưa được phát).

Hiện tại, *<audio>*/*<video>* chỉ hỗ trợ các định dạng âm thanh MP3, WAV, OGG, cùng các định dạng phim MP4, WebM và OGG. Tuy nhiên, thực tế còn có có nhiều định dạng âm thanh và phim khác. Do vậy, để có một trình phát đa phương tiện (media player) trên web có thể phát nhiều định dạng âm thanh và phim, nhiều đối tượng nhúng phải được dùng kết hợp và được điều phối, điều khiển bởi JavaScript. *MediaElement.js*[7] là một ví dụ về trình phát đa phương tiện trên web.

2.14.4. Khung nội tuyến

Đối tượng *<iframe>* (inline frame) được thiết kế chuyên biệt cho mục đích nhúng tài liệu HTML vào tài liệu HTML khác. Mặc dù có thể sử dụng *<object>* cho mục đích này, *<iframe>* có sự khác biệt căn bản so với *<object>* đó là *<iframe>* khai báo một cửa sổ/ngữ cảnh con được lồng (nested) trong cửa sổ/ngữ cảnh cha. Nếu

[7] http://www.mediaelementjs.com/

chỉ cần hiển thị tài liệu HTML được nhúng, sử dụng <*object*> hay <*iframe*> đều được. Tuy nhiên, nếu cần tương tác[8] giữa các đối tượng thuộc tài liệu nhúng với các đối tượng thuộc tài liệu bên ngoài thì đối tượng <*iframe*> cần được sử dụng. Ví dụ sử dụng <*iframe*> để nhúng tài liệu như sau.

```
1. <iframe src="wall.htm"></iframe>
```

Ngoài thuộc tính *src* cho biết URL của tài liệu HTML được nhúng, đối tượng <*iframe*> còn có các thuộc tính *width* và *height* để xác định độ rộng và chiều cao của cửa sổ nhúng khi được hiển thị trên giao diện của trình duyệt.

2.15. TRÌNH BÀY, NHÓM GỘP

2.15.1. Ngắt dòng hiển thị

Đối tượng <*br*> (break) có tác dụng ngắt dòng hiển thị. Đối tượng sau <*br*> sẽ được hiển thị trên dòng mới. Ví dụ, ba đối tượng văn bản sau đây được hiển thị trên ba dòng do có các đối tượng <*br*> nằm giữa chúng.


```
1. Tre xanh<br>xanh tự bao giờ <br>Chuyện ngày xưa đã có bờ tre xanh?
```

2.15.2. Nhóm gộp

Thông thường, các đối tượng tài liệu nên được gộp vào các nhóm để dễ quản lý và trình diễn. Các đối tượng <*span*> và <*div*> (division) được sử dụng cho mục đích này. <*span*> được sử dụng để nhóm gộp các đối tượng hiển thị theo dòng (inline elements) trong khi <*div*> được sử dụng để nhóm gộp các đối tượng hiển thị theo khối (block elements)[9]. <*div*> được sử dụng rất nhiều, cùng với CSS, trong việc tạo bố cục (layout) hay dàn trang. <*span*> cũng có thể được sử dụng cùng CSS để tạo kiểu dáng cho các đối tượng bên trong nó. Ví dụ sau khai báo các đối tượng <*span*> và <*div*>.

```
1. <div>
2.   <h3>Thông báo tạm ngừng cấp điện</h3>
3.   <p><span style="color:red;">Công ty Điện lực Ba Đình</span> xin thông
   báo về việc tạm thời cắt điện …</p>
4. </div>
```

2.16. NHẬP LIỆU

Các ứng dụng web không chỉ cung cấp nội dung một chiều từ ứng dụng đến trình duyệt mà còn có thể nhận dữ liệu được gửi từ trình duyệt đến và xử lý dữ liệu nhận được. Để cung cấp dữ liệu cho ứng dụng, từ giao diện của trình duyệt, cụ thể là trên thanh nhập địa chỉ của trình duyệt, người dùng có thể đưa các tham số vào chuỗi truy vấn trong URL. Tuy nhiên, cách nhập liệu như vậy không mấy

[8] Sử dụng JavaScript để thực hiện tương tác.
[9] Các kiểu hiển thị theo dòng và theo khối được trình bày trong Chương 3, Mục 3.8.

thuận tiện. Cách nhập liệu thuận tiện hơn là cung cấp một giao diện trực quan trên đó người dùng được hướng dẫn cách nhập và có thể nhập một cách dễ dàng. Các đối tượng nhập liệu được thiết kế để thực hiện chức năng này, bao gồm *<input>*, *<textarea>* và *<select>*. Ngoài ra, các đối tượng *<form>* và *<button>* được sử dụng để kết thúc nhập liệu và đệ trình dữ liệu cho trình phục vụ.

2.16.1. Dữ liệu văn bản

Đối tượng *<textarea>* được sử dụng để nhập văn bản. Thông qua đối tượng này, người dùng có thể nhập văn bản với số dòng và số ký tự không giới hạn. Để chỉ định vùng nhìn thấy của văn bản được nhập, các thuộc tính *cols* và *rows* được sử dụng. Nếu không nhìn thấy toàn bộ văn bản, trình duyệt sẽ tự động tạo thanh cuộn cho vùng nhập. Ví dụ khai báo đối tượng *<textarea>* như sau.

```
1. <textarea rows="4" cols="50" placeholder="Mô tả sở thích của bạn">Bóng đá,
    quần vợt và nhạc trữ tình.</textarea>
```

Văn bản khởi tạo cho *<textarea>* được đặt giữa thẻ mở và thẻ đóng. Người dùng có thể nhập tiếp và/hoặc chỉnh sửa văn bản khởi tạo. Dĩ nhiên, văn bản khởi tạo có thể rỗng. Nếu văn bản của *<textarea>* là rỗng, giá trị của thuộc tính *placeholder* sẽ được hiển thị trong vùng nhập. Giá trị này có tác dụng hướng dẫn hoặc gợi ý cho người dùng cách nhập.

2.16.2. Dữ liệu kiểu liệt kê

Nếu dữ liệu được chấp nhận bởi ứng dụng chỉ thuộc một danh sách được xác định trước, hay còn gọi là dữ liệu có kiểu liệt kê (enum), đối tượng *<select>* là lựa chọn phù hợp nhất cho việc nhập liệu. Bản thân *<select>* chỉ là đối tượng chứa. Các mục dữ liệu được khai báo bằng các đối tượng *<option>* nằm trong *<select>*. Ngoài ra, đối tượng *<select>* có thể, nhưng không bắt buộc, chứa các đối tượng *<optgroup>* để gộp các *<option>* thành các nhóm logic. Ví dụ sau khai báo một đối tượng *<select>*.

```
1. <select>
2.   <optgroup label="Châu Á">
3.     <option value="vn">Việt Nam</option>
4.     <option value="jp">Nhật Bản</option>
5.     <option value="kr">Hàn Quốc</option>
6.   </optgroup>
7.   <optgroup label="Châu Âu">
8.     <option value="en">Anh</option>
9.     <option value="fr">Pháp</option>
10.  </optgroup>
11. </select>
```

Trong ví dụ này, người dùng chỉ được chọn một trong năm giá trị là vn, jp, kr, en và fr; mỗi giá trị được xác định bằng thuộc tính *value* của một đối tượng *<option>*. Lưu ý rằng nội dung văn bản nằm giữa thẻ mở và thẻ đóng *<option>* chỉ là hiển thị

bên ngoài giúp người dùng quan sát, giá trị của thuộc tính *value* mới là dữ liệu sẽ được đệ trình cho trình phục vụ.

Mặc định đối tượng *<select>* chỉ cho chọn một giá trị duy nhất. Để đối tượng *<select>* cho phép chọn nhiều giá trị đồng thời, thuộc tính *multiple* của *<select>* được sử dụng. Khi có mặt thuộc tính này, trình duyệt sẽ hiển thị *<select>* dưới dạng một khung có thanh cuộn thay vì một danh sách thả xuống như mặc định.

2.16.3. Dữ liệu tùy biến

Đa số các tình huống sử dụng yêu cầu/cho phép người dùng nhập các dữ liệu tùy biến (có giá trị bất kỳ, không bị ràng buộc trong một danh sách cho trước). Đối tượng *<input>* được thiết kế cho mục đích này. Dữ liệu tùy biến được nhập vào *<input>* sẽ được đệ trình cho trình phục vụ như một xâu ký tự. Tuy nhiên, để giúp người dùng dễ dàng hơn trong thao tác nhập liệu, đối tượng *<input>* có nhiều biến thể trong trình diễn và tương tác với người dùng. Các biến thể được xác định bằng thuộc tính *type* của nó. Ở dạng đơn giản nhất, *<input>* được thể hiện như một ô nhập chữ (*type="text"*), người dùng có thể nhập một xâu ký tự bất kỳ (nhưng không có ký tự xuống dòng). Ở các dạng khác, *<input>* được thể hiện như một hộp kiểm (*type="checkbox"*), một nút radio (*type="radio"*), một hộp chọn màu (*type="color"*), hộp chọn ngày tháng (*type="date"*), một hộp chọn tệp (*type="file"*), hay một thanh trượt để chọn một cấp độ thuộc một khoảng cho trước (*type="range"*). Các biến thể còn lại bao gồm *button, datetime-local, email, hidden, image, month, number, password, reset, search, submit, tel, time, url,* và *week*. Tùy theo biến thể, *<input>* có thêm một vài thuộc tính riêng khác. Ví dụ, các biến thể *radio* và *checkbox* có thuộc tính *checked* cho biết nút được chọn hay không, các biến thể *number* và *date* có các thuộc tính *min* và *max* để xác định giá trị nhỏ nhất và giá trị lớn nhất người dùng có thể nhập. Ngược lại, dù ở biến thể nào, giá trị được nhập vào *<input>* cũng được gán cho thuộc tính *value* của nó. Ví dụ đoạn mã sau khai báo ba đối tượng *<input>*, trong đó có một biến thể *text* và hai biến thể *radio*. Lưu ý, để các *radio* thuộc về cùng một nhóm loại trừ (nếu chọn phần tử này sẽ bỏ chọn các phần tử còn lại trong nhóm), chúng phải có *name* như nhau.


```
1. Họ và tên:
2. <input type="text" name="fullname"><br>
3. Giới tính:
4. <input  type="radio" value="M" name="gender"> Nam
5. <input  type="radio" value="F" name="gender"> Nữ
```

2.16.4. Đệ trình dữ liệu

Các đối tượng nhập liệu đã được trình bày ở trên có chức năng tạo giao diện cho người dùng nhập, đồng thời nắm giữ dữ liệu người dùng đã nhập. Tuy nhiên, các đối tượng này không thể tự gửi dữ liệu đến trình phục vụ được. Để thực hiện điều này, đối tượng *<form>* phải được khai báo, đồng thời các đối tượng nhập liệu phải được đặt trong *<form>* trước khi *<form>* được đệ trình (submit). Một cách ví

von, *<form>* giống như máy bay, các đối tượng nhập liệu giống như hành khách và hàng hóa được chở trên máy bay, trình duyệt giống như cảng đi, trình phục vụ giống như cảng đến của máy bay; tất cả hành khách và hàng hóa trên cùng một máy bay sẽ được di chuyển và đến đích đồng thời; hành khách và hàng hóa không ở trên máy bay sẽ không được di chuyển.

Trở lại với đối tượng *<form>*, một trong những thuộc tính quan trọng của nó là *action*. Thuộc tính này cho biết URL của tài nguyên được gọi khi đệ trình *<form>*. Tài nguyên đó sẽ tiếp nhận và xử lý dữ liệu được đệ trình, cũng như trả nội dung kết quả về cho trình duyệt. Nếu *action* vắng mặt hoặc giá trị của nó là rỗng, tài nguyên được triệu gọi chính là trang hiện tại. Một thuộc tính nữa cũng thường xuyên được sử dụng là *method*. Nó cho biết phương thức HTTP sẽ được sử dụng khi đệ trình *<form>*. Giá trị của *method* có thể là GET (mặc định) hoặc POST. Thông thường, POST được khuyến cáo sử dụng, đặc biệt trong các tình huống cần đệ trình nhiều dữ liệu đồng thời, dữ liệu có dung lượng lớn. Các thuộc tính khác như *enctype*, *target* ít được dùng hơn. *enctype* cho biết phương thức biểu diễn dữ liệu trước khi đệ trình. *target* cho biết cửa sổ sẽ hiển thị tài nguyên được trả về. Giá trị của *target* có thể là *_blank* (cửa sổ, tab mới), *_self* (cửa sổ hiện tại), *_parent* (cửa sổ cha), *_top* (cửa sổ cao nhất hay là cửa sổ trình duyệt).

Có hai sự kiện cơ bản trên *<form>* là đệ trình (submit – gọi đến *action* và gửi dữ liệu đi) và làm lại (reset – xóa trắng dữ liệu đã nhập trên các đối tượng nhập liệu thuộc *<form>*). Cả hai sự kiện này đều được thực hiện bằng cách đưa các biến thể *<input>* tương ứng (*submit* hay *reset*) vào trong *<form>* để người dùng kích hoạt sự kiện bằng cách bấm vào các nút. Ví dụ sau là một khai báo *<form>* khá điển hình.

```
1. <form action="/add.php" method="post">
2.   Họ tên: <input type="text" name="fullname"><br>
3.   Ngày sinh: <input type="text" name="dob"><br>
4.   <input type="submit" value="Chấp nhận">
5. </form>
```

2.16.5. Hỗ trợ nhập liệu

Bên cạnh các đối tượng nhập liệu và đệ trình dữ liệu, một số đối tượng hỗ trợ nhập liệu cũng thường xuyên được sử dụng. Đối tượng hỗ trợ nhập liệu thứ nhất là *<label>*, hay là nhãn. Nhãn được đặt bên cạnh, trước hoặc sau, đối tượng nhập liệu, có tác dụng chú thích cho dữ liệu nhập. Đồng thời, nhãn cung cấp một tiện ích nhỏ để khi người dùng kích chuột vào nhãn, tâm điểm sẽ được đặt vào đối tượng nhập liệu được nhãn bổ trợ. Ví dụ, *<form>* được khai báo ở ví dụ trước được chỉnh sửa, thay các đối tượng văn bản bằng các đối tượng nhãn, như sau. Thuộc tính *for* của nhãn có giá trị là *id* của đối tượng nhập liệu mà nó bổ trợ.

```
1. <form action="/add.php" method="post">
2.   <label for="fname">Họ tên:</label>
3.   <input type="text" name="fullname" id="fname">
4.   <br>
5.   <label>Ngày sinh:<label>
```

```
6.    <input type="text" name="dob">
7.    <br>
8.    <input type="submit" value="Chấp nhận">
9.  </form>
```

Đối tượng hỗ trợ thứ hai là *<fieldset>*. Nếu như *<label>* chú thích cho từng đối tượng nhập liệu thì *<fieldset>* chú thích cho cả nhóm đối tượng nhập liệu. Ví dụ, *<form>* ở trên được bổ sung chú thích nhóm như sau.

```
1.  <form action="/add.php" method="post">
2.    <fieldset>
3.      <legend>Thông tin cá nhân:</legend>
4.      <label for="fname">Họ tên:</label>
5.      <input type="text" name="fullname" id="fname"><br>
6.      <label>Ngày sinh:<label>
7.      <input type="text" name="dob">
8.    </fieldset>
9.    <input type="submit" value="Chấp nhận">
10. </form>
```

Đối tượng hỗ trợ tiếp theo là *<datalist>*. Khác với *<label>* và *<fieldset>* có tác dụng về mặt trình bày, *<datalist>* cung cấp khả năng tự động hoàn thành (autocomplete) cho *<input>*. Khi người dùng kích chuột vào *<input>* có *<datalist>*, các giá trị của *<datalist>* sẽ được hiển thị để người dùng chọn một trong các giá trị đó. Một ví dụ sử dụng *<datalist>* như sau.


```
1.  <input list="browsers">
2.  <datalist id="browsers">
3.    <option value="Firefox">
4.    <option value="Chrome">
5.    <option value="Opera">
6.  </datalist>
```

2.17. CẬP NHẬT KIẾN THỨC VỀ HTML

Nội dung chương đã giới thiệu và trình bày những đối tượng tài liệu thường xuyên được sử dụng nhất. Những đối tượng tài liệu ít được sử dụng hơn không được trình bày trong giáo trình này do giới hạn dung lượng của giáo trình. Bạn đọc quan tâm các đối tượng kể trên có thể tìm đọc ở các tài liệu khác. Ngoài ra, đặc tả HTML liên tục được W3C nâng cấp, sửa đổi (phiên bản hiện tại là 5). Một số đối tượng hay thuộc tính đang có sẽ trở nên lỗi thời trong khi một số đối tượng hay thuộc tính mới sẽ được bổ sung. Vì vậy, việc nắm vững vai trò khai báo ra đối tượng tài liệu của HTML cùng với việc cập nhật kiến thức về các phiên bản HTML là yêu cầu bắt buộc đối với các lập trình viên web.

Bài tập

1. Tạo một trang HTML, nhập các đối tượng tài liệu được khai báo trong các ví dụ được trình bày ở chương này vào trang HTML vừa tạo, nhập xong mỗi ví

dụ thì tải lại trang bằng trình duyệt để thấy thể hiện của các đối tượng trên giao diện.

2. Tạo một trang HTML có một bảng hiển thị danh sách các nhân viên của một công ty, mỗi nhân viên có các thông tin về họ tên, ngày sinh, vị trí công việc đang đảm nhiệm, số điện thoại và email liên hệ.
3. Tạo một trang HTML khác có một bài viết giới thiệu về công ty và một liên kết đến trang hiển thị danh sách nhân viên của công ty.

Đọc thêm

1. ClydeBank Technology, Martin Mihajlov, "HTML QuickStart Guide: The Simplified Beginner's Guide To HTML", ClydeBank Media LLC, 2015.
2. Gilad E. Tsur Mayer, "HTML: HTML Awesomeness Book - Learn Write HTML The Awesome Way", CreateSpace Independent Publishing Platform, 2016.
3. Michael Abelar, "Ultra HTML Reference: An in-depth reference book for the HTML programming language", Amazon Digital Services LLC, 2016.

Chương 3

ĐỊNH KIỂU TRÌNH DIỄN TRANG WEB BẰNG CSS

3.1. BẢNG ĐỊNH DẠNG

CSS (Cascading Style Sheet) là ngôn ngữ thứ hai được sử dụng để tạo ra nội dung web. Nếu như HTML được sử dụng để khai báo ra các đối tượng tài liệu thì CSS được sử dụng để xác định kiểu trình diễn các đối tượng tài liệu. Sử dụng CSS không chỉ nhằm tách biệt nội dung với trình diễn mà còn nhằm đảm bảo sự nhất quán trong trình diễn cũng như có thể sử dụng lại mã trong trình diễn. CSS cung cấp duy nhất một phần tử là bảng định dạng (style sheet). Bảng định dạng cho biết áp dụng kiểu dáng gì trên những đối tượng tài liệu nào. Cú pháp khai báo bảng định dạng như sau:

Bộ_chọn {thuộc_tính_1: giá_trị_1; thuộc_tính_2: giá_trị_2; ...}

trong đó, *Bộ_chọn* (selector) cho biết những đối tượng tài liệu được áp dụng kiểu dáng xác định trong thân (body) của bảng định dạng, và thân của bảng định dạng đơn giản là danh sách các cặp *thuộc_tính:giá_trị* được phân cách nhau bởi dấu chấm phẩy (;) và đặt giữa các dấu ngoặc mở ({), ngoặc đóng (}). Thuộc tính được liệt kê trong thân của bảng định dạng chỉ liên quan đến kiểu dáng (style) như màu sắc, kích thước, phông chữ, … và không liên quan đến nội dung. Bộ chọn chỉ rơi vào một số lớp xác định trước, được trình bày trong mục tiếp theo.

3.2. BỘ CHỌN

Cơ bản, CSS có hai bộ chọn là chọn theo phần tử và chọn theo thuộc tính. Bộ chọn theo phần tử chọn tất cả các đối tượng tài liệu được khai báo bởi cùng một thẻ HTML, trong khi bộ chọn theo thuộc tính chọn tất cả các đối tượng tài liệu có một thuộc tính được chỉ định. Bộ chọn theo thuộc tính có những thể hiện chi tiết hơn như chọn theo định danh (thuộc tính *id*), chọn theo lớp (thuộc tính *class*).

3.2.1. Bộ chọn theo phần tử

Bộ chọn theo phần tử có tên trùng với tên thẻ HTML, có tác dụng chọn các đối tượng được khai báo bởi thẻ là tên bộ chọn. Có bao nhiêu thẻ HTML thì có bấy nhiêu bộ chọn theo phần tử, tương ứng. Trong trang web ví dụ sau đây, bảng định dạng với bộ chọn *label* làm cho hai đối tượng nhãn có nội dung lần lượt là "Đây là nhãn" và "Đây là nhãn khác" được hiển thị với chữ màu đỏ, trong khi đoạn văn với nội dung "Đây là đoạn văn" cùng đối tượng văn bản với nội dung "Đây là văn bản thường" được hiển thị với chữ màu đen theo mặc định.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.2.1</title>
3.   <meta charset="utf-8">
4.   <style type="text/css">
5.     label {color:red;}
6.   </style>
7. </head><body>
8.   <label>Đây là nhãn</label>
9.   <label>Đây là nhãn khác</label>
10.  <p>Đây là đoạn văn</p>
11.  Đây là văn bản bình thường
12. </body></html>
```

3.2.2. Bộ chọn theo thuộc tính

Bộ chọn theo thuộc tính có dạng [*att*] hoặc [*att* = *v*], [*att* *= *v*], [*att* ~= *v*], [*att* |= *v*], trong đó *att* và *v* là các xâu ký tự, có tác dụng chọn các đối tượng tài liệu có thuộc tính *att* hoặc có thuộc tính *att* với giá trị là *v*, có thuộc tính *att* với giá trị chứa *v*, có thuộc tính *att* với giá trị chứa từ *v*, có thuộc tính *att* với giá trị bắt đầu bằng từ *v*, tương ứng (từ ở đây được hiểu là chuỗi ký tự không bao gồm dấu trắng và dấu gạch ngang (-)). Trong trang web ví dụ sau đây, bảng định dạng thứ nhất với bộ chọn [*href*] làm cho cả ba liên kết được hiển thị trên trình duyệt với chữ màu xanh lá, bảng định dạng thứ hai với bộ chọn [*href* *= *".edu"*] làm cho liên kết đầu tiên được hiển thị với chữ đậm, và bảng định dạng cuối cùng với bộ chọn [*lang* |= *"en"*] làm cho liên kết thứ ba được hiển thị với chữ nghiêng.


```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.2.2</title>
3.   <meta charset="utf-8">
4.   <style type="text/css">
5.     [href] {color:green;}
6.     [href *= ".edu"] {font-weight:bold;}
7.     [lang |= "en"] {font-style:italic;}
8.   </style>
9. </head><body>
10.  <h2>Một số liên kết</h2>
11.  <a href="http://vnu.edu.vn" lang=vi>Đại học Quốc gia Hà Nội</a>
12.  <a href="http://dantri.com.vn" lang=vi>Báo điện tử Dân trí</a>
13.  <a href="https://google.com" lang="en-us">Google</a>
14. </body></html>
```

3.2.3. Bộ chọn theo định danh

Bộ chọn theo định danh có dạng #*v*, có tác dụng chọn đối tượng có định danh (id) là *v*. Về mặt ngữ nghĩa, #*v* tương đương [*id*=*v*], hay bộ chọn theo định danh là trường hợp riêng của bộ chọn theo thuộc tính. Bộ chọn theo định danh rất hay được sử dụng trong các trang web. Trong trang web ví dụ sau đây, bảng định dạng với bộ chọn #*vnu* làm cho liên kết đầu tiên được hiển thị với chữ không có gạch chân, trong khi các liên kết còn lại được hiển thị với chữ có gạch chân theo mặc định.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.2.3</title>
3.   <meta charset="utf-8">
4.   <style type="text/css">
5.     #vnu {text-decoration:none;}
6.   </style>
7. </head><body>
8.   <h2>Một số liên kết</h2>
9.   <a href="http://vnu.edu.vn" id="vnu">Đại học Quốc gia Hà Nội</a>
10.  <a href="http://dantri.com.vn" id="dantri">Báo điện tử Dân trí</a>
11.  <a href="https://google.com" id="google">Google</a>
12. </body></html>
```

3.2.4. Bộ chọn theo lớp

Bộ chọn theo lớp có dạng *.v*, có tác dụng chọn đối tượng có thuộc tính *class* với giá trị là *v*. Về mặt ngữ nghĩa, *.v* tương đương [*class=v*], hay bộ chọn theo lớp cũng là trường hợp riêng của bộ chọn theo thuộc tính. Bộ chọn theo lớp được dùng nhiều nhất trong các trang web. Trong trang web ví dụ sau đây, bảng định dạng với bộ chọn *.important* làm cho liên kết đầu tiên được hiển thị với cỡ chữ lớn hơn, trong khi các liên liên kết còn lại được hiển thị với cỡ chữ mặc định.


```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.2.4</title>
3.   <meta charset="utf-8">
4.   <style type="text/css">
5.     .important {font-size:x-large;}
6.   </style>
7. </head><body>
8.   <h2>Một số liên kết</h2>
9.   <a href="http://vnu.edu.vn" class="important">ĐHQGHN</a>
10.  <a href="http://dantri.com.vn" class="normal">Báo điện tử Dân trí</a>
11.  <a href="https://google.com" class="normal">Google</a>
12. </body></html>
```

3.2.5. Bộ chọn nội tuyến

Thuộc tính *style*, còn được gọi là bộ chọn nội tuyến (inline selector), được khai báo cùng khai báo đối tượng HTML. Giá trị gán cho *style* giống như nội dung thân của các bộ chọn đã trình bày ở trước. Trong trang web ví dụ sau đây, liên kết đầu tiên được định kiểu, với chữ màu đỏ, bằng thuộc tính *style*.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.2.5</title>
3.   <meta charset="utf-8">
4. </head><body>
5.   <h2>Một số liên kết</h2>
6.   <a href="http://vnu.edu.vn" style="color:red;">ĐHQGHN</a>
7.   <a href="http://dantri.com.vn">Báo điện tử Dân trí</a>
8.   <a href="https://google.com">Google</a>
9. </body></html>
```

3.2.6. Bộ chọn tất cả

Bộ chọn tất cả, ký hiệu là *, có tác dụng chọn tất cả các đối tượng tài liệu trong trang web. Bộ chọn này ít được sử dụng. Trong trang web ví dụ sau đây, tất cả chữ được hiển thị trên giao diện đều có màu đỏ.

```
1.  <!DOCTYPE html><html><head>
2.    <title>L.3.2.6</title>
3.    <meta charset="utf-8">
4.    <style type="text/css">
5.      * {color:red;}
6.    </style>
7.  </head><body>
8.    <h2>Một số liên kết</h2>
9.    <a href="http://vnu.edu.vn" id="vnu">Đại học Quốc gia Hà Nội</a>
10.   <a href="http://dantri.com.vn" class="normal">Báo điện tử Dân trí</a>
11.   <a href="https://google.com">Google</a>
12. </body></html>
```

3.2.7. Lớp giả, phần tử giả

Lớp giả (pseudo class) và phần tử giả (pseudo element) có thể được thêm vào sau các bộ chọn để thay đổi điều kiện hay phạm vi chọn các đối tượng tài liệu. Dưới đây là một số lớp giả hay được sử dụng:

:hover	Chọn đối tượng chỉ khi nó được đưa chuột vào.
:focus	Chọn đối tượng chỉ khi nó được đặt tâm điểm.
:link	Chọn đối tượng liên kết chỉ khi nó chưa được thăm.
:visited	Chọn đối tượng liên kết chỉ khi nó đã được thăm.
:active	Chọn đối tượng liên kết chỉ khi nó đang được thăm.
:first-child	Chọn đối tượng chỉ khi nó là con đầu tiên của một đối tượng khác.
:last-child	Chọn đối tượng chỉ khi nó là con cuối cùng của một đối tượng khác.
:nth-child(n)	Chọn đối tượng chỉ khi nó là con thứ *n* của một đối tượng khác.
:only-child	Chọn đối tượng chỉ khi nó là con duy nhất của mộ đối tượng khác.
:empty	Chọn đối tượng chỉ khi nó không có đối tượng con.
:disabled	Chọn đối tượng chỉ khi nó đã bị vô hiệu hóa.

Dưới đây là các phần tử giả:

::first-letter	Chọn ký tự đầu tiên thuộc văn bản bên trong đối tượng được chọn.

::*first-line* Chọn dòng đầu tiên thuộc văn bản bên trong đối tượng được chọn.

::*before* Được sử dụng để chèn ngữ cảnh nào đó vào trước đối tượng được chọn.

::*after* Được sử dụng để chèn ngữ cảnh nào đó vào sau đối tượng được chọn.

::*selection* Chọn một phần của đối tượng được người dùng chọn bằng giữ và kéo chuột.

Lưu ý, tên của các lớp giả có dấu hai chấm (:) ở đầu trong khi tên của các phần tử giả có hai dấu hai chấm (::) ở đầu. Trong trang web ví dụ sau đây, sau mỗi liên kết đều được chèn thêm ba dấu gạch ngang (---) do tác dụng của bảng định dạng thứ nhất, và liên kết đầu tiên được hiển thị với chữ màu đỏ do tác dụng của bảng định dạng thứ hai.

```
1.  <!DOCTYPE html><html><head>
2.    <title>L.3.2.7</title>
3.    <meta charset="utf-8">
4.      <style type="text/css">
5.        a::after {content:"---";}
6.        a:first-child {color:red;}
7.      </style>
8.  </head><body>
9.    <div>
10.     <a href="http://vnu.edu.vn">Đại học Quốc gia Hà Nội</a>
11.     <a href='http://dantri.com.vn'>Báo điện tử Dân trí</a>
12.   </div>
13.   <a href ='https://google.com'>Google</a>
14. </body></html>
```

3.2.8. Kết hợp nhiều bộ chọn

Có thể sử dụng kết hợp các bộ chọn để thay đổi phạm vi hay điều kiện chọn các đối tượng tài liệu. Dưới đây là một số kết hợp phổ biến:

p > *e* Chọn đối tượng được chọn bởi *e* có cha được chọn bởi *p*.

a e Chọn đối tượng được chọn bởi *e* ở bên trong, hay có tổ tiên là đối tượng được chọn bởi *a*.

prev + *e* Chọn đối tượng được chọn bởi *e* có đối tượng liền trước được chọn bởi *prev*.

prev ~ *e* Chọn đối tượng được chọn bởi *e* nằm sau đối tượng được chọn bởi *prev*.

Trong trang web ví dụ sau đây, hai liên kết đầu tiên được hiển thị với chữ màu đỏ do được áp dụng bảng định dạng thứ nhất, liên kết đầu tiên còn có chữ in đậm do được áp dụng bảng định dạng thứ hai, liên kết thứ ba được hiển thị với chữ màu

xanh lá cây do được áp dụng bảng định dạng thứ ba, trong khi liên kết cuối cùng được hiển thị với chữ màu xanh da trời theo mặc định.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.2.8</title>
3.   <meta charset="utf-8">
4.     <style type="text/css">
5.       div a {color:red;}
6.       div>a {font-weight:bold;}
7.       div+a {color:green;}
8.     </style>
9. </head><body>
10.   <div>
11.     <h2>Một số liên kết</h2>
12.     <a href="http://vnu.edu.vn">Đại học Quốc gia Hà Nội</a>
13.     <span><a href='http://dantri.com.vn'>Báo điện tử Dân trí</a></span>
14.   </div>
15.   <a href ='https://google.com'>Google</a>
16.   <a href='http://vnexpress.net'>VnExpress</a>
17. </body></html>
```


3.2.9. Viết gộp nhiều bộ chọn

Nếu có nhiều bảng định dạng mà thân của chúng có những thuộc tính chung, các bộ chọn có thể được viết gộp để tránh phải viết lại mã. Các thuộc tính chung xuất hiện trong lần viết gộp, trong khi những thuộc tính riêng xuất hiện tại những bộ chọn riêng rẽ, như ví dụ sau đây.

```
1. h1, h2, p {
2.   font-family:Arial;
3.   color:red;
4. }
5. p {text-align:justify;}
6. h1 {text-decoration:underline;}
```

3.3. KHAI BÁO CSS

Các bảng định dạng có thể được khai báo bằng đối tượng *<style>* trong phần đầu (head) như đã thấy qua các ví dụ ở trên. Mã nguồn CSS được khai báo theo cách này được gọi là *CSS trong* (internal CSS). CSS trong chỉ có tác dụng cho trang web chứa nó và không thể sử dụng lại cho các trang khác. Để khắc phục điều đó, CSS có thể được viết ở một tệp riêng biệt và trang web bất kỳ có thể bao hàm (include) tệp CSS. CSS ở tệp riêng biệt được gọi là *CSS ngoài* (external CSS). Tệp chứa CSS ngoài có tên mở rộng là *.css*. Bao hàm CSS ngoài được thực hiện bằng đối tượng *<link>* như đã được trình bày trong Mục 2.8. Cụ thể, giả sử các bảng định dạng được định nghĩa ở tệp *mystyle.css*, trang web cần bao hàm tệp *mystyle.css* sẽ khai báo đối tượng *<link>* như sau trong phần đầu.

```
1. <link rel="stylesheet" type="text/css" href="mystyle.css"/>
```

3.4. CHÚ THÍCH TRONG CSS

Chú thích trong CSS được đặt giữa /* và */ tương tự chú thích trên nhiều dòng trong C++.

```
1. /* Đây là chú thích
2. trong CSS */
```

3.5. BẢNG ĐỊNH DẠNG KẾ THỪA VÀ MẶC ĐỊNH

Khi một đối tượng tài liệu được chọn bởi một bộ chọn, tất cả các đối tượng bên trong nó cũng được chọn theo. Nói cách khác, đối tượng con được kế thừa các bảng định dạng từ đối tượng cha. Ngoài ra, mọi đối tượng tài liệu đều được áp dụng bảng định dạng mặc định theo phần tử của trình duyệt. Nhờ có các bảng định dạng mặc định theo phần tử của trình duyệt mà các đối tượng tài liệu được trình diễn ngay cả khi CSS không được khai báo trong trang web. Ở trang web ví dụ sau đây, hai đối tượng *<label>* đầu tiên được kế thừa bảng định dạng từ đối tượng * * nên có chữ màu đỏ, trong khi đối tượng *<label>* còn lại có chữ màu đen theo mặc định.


```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.5</title>
3.   <style> div {color:red;} </style>
4. </head><body>
5.     <div><label>Nhãn 1<label></div>
6.     <div><span><label>Nhãn 2</label></span></div>
7.     <label>Nhãn 3</label>
8. </body></html>
```

3.6. THỨ TỰ ƯU TIÊN CÁC BẢNG ĐỊNH DẠNG

Khi có nhiều bảng định dạng cùng chọn một đối tượng tài liệu, đối tượng tài liệu sẽ chịu tác động của tất cả các bảng định dạng chọn nó. Tuy nhiên, xung đột có thể xảy ra khi hai bảng định dạng cùng chứa một thuộc tính mà giá trị của chúng khác nhau ở hai bảng. Khi đó, trình duyệt sẽ phải sử dụng thứ tự ưu tiên (precedence) của các bảng định dạng để chọn giá trị trong bảng định dạng có độ ưu tiên cao hơn. Ba quy tắc xác định độ ưu tiên cho các bảng định dạng đã được đưa ra. Các quy tắc này sẽ được trình bày ngay sau đây. Thứ tự trình bày các quy tắc cũng là thứ tự áp dụng chúng. Nếu đã sử dụng quy tắc trước mà vẫn không phân định được cao-thấp các độ ưu tiên, tức các độ ưu tiên bằng nhau, thì áp dụng tiếp quy tắc liền sau.

Quy tắc thứ nhất căn cứ vào tính **kế thừa** (inheritance) của các bảng định dạng. Theo quy tắc này, bảng định dạng được kế thừa gần hơn sẽ có độ ưu tiên cao hơn. Bảng định dạng được áp dụng trên chính đối tượng có độ ưu tiên cao nhất, sau đó mới đến bảng định dạng kế thừa từ đối tượng cha, bảng định dạng kế thừa từ đối tượng ông, … Bảng định dạng mặc định (kế thừa xa nhất) có độ ưu tiên thấp nhất.

Quy tắc thứ hai căn cứ vào tính **cụ thể** (specificity) của bộ chọn, tức mức độ rõ ràng trong việc chỉ định đích danh các đối tượng tài liệu được chọn. Các bộ chọn khác nhau có tính cụ thể khác nhau. Bộ chọn có tính cụ thể càng cao thì thứ tự ưu tiên của bảng định dạng càng cao. Ví dụ, bộ chọn theo định danh chỉ đích danh một đối tượng được chọn trong khi bộ chọn theo phần tử chỉ xác định một nhóm các đối tượng, do vậy tính cụ thể và độ ưu tiên của bộ chọn theo định danh sẽ cao hơn tính cụ thể và độ ưu tiên của bộ chọn theo phần tử. Tính cụ thể của mỗi bộ chọn được lượng hóa bằng một vector bốn chiều. Tính cụ thể của kết hợp các bộ chọn được xác định bằng vector tổng của các vectors là tính cụ thể của các bộ chọn thành phần. So sánh tính cụ thể được thực hiện bằng so sánh vector, tức là lần lượt so sánh các các phần tử trong vector từ trái qua phải. Dưới đây là danh sách các bộ chọn theo thứ tự giảm dần tính cụ thể:

Bộ chọn	**Tính cụ thể**
Nội tuyến	<1, 0, 0, 0>
Theo định danh	<0, 1, 0, 0>
Theo thuộc tính, theo lớp, lớp giả	<0, 0, 1, 0>
Theo phần tử, phần tử giả	<0, 0, 0, 1>
Tất cả (*)	<0, 0, 0, 0>

Dễ dàng tính được tính cụ thể của kết hợp các bộ chọn bằng cách cộng các vectors. Ví dụ, kết hợp "*ul#nav*" có tính cụ thể là <0, 1, 0, 1>, kết hợp "*ul#nav li.active a*" có tính cụ thể là <0, 1, 1, 3>. Tương tự, dễ dàng so sánh được các vector, tức so sánh được các tính cụ thể với nhau.

Quy tắc thứ ba căn cứ vào **thứ tự nối** (cascading) các bảng định dạng. Theo quy tắc này, bảng định dạng được khai báo/nối sau sẽ có độ ưu tiên cao hơn. Nói cách khác, thuộc tính khai báo sau sẽ ghi đè lên cùng thuộc tính được khai báo trước. Tuy nhiên, nhắc lại và lưu ý rằng quy tắc thứ ba chỉ được áp dụng sau quy tắc thứ nhất và quy tắc thứ hai.

Trong trang web ví dụ sau đây, ba đối tượng <*label*> được khai báo. Đối tượng thứ nhất được hai bộ chọn *label* cùng chọn, trong đó một bộ chọn từ CSS ngoài và một bộ chọn từ CSS trong. Vị trí bao hàm CSS ngoài ở sau CSS trong nên bộ chọn *label* từ CSS ngoài có độ ưu tiên cao hơn. Quy tắc thứ ba đã được áp dụng trên đối tượng <*label*> thứ nhất dẫn đến đối tượng <*label*> thứ nhất có chữ màu vàng. Đối tượng <*label*> thứ hai được chọn bởi hai bộ chọn *label* và bộ chọn *label.note*. Bộ chọn *label.note* có tính cụ thể cao hơn bộ chọn *label*. Quy tắc thứ hai đã được áp dụng trên đối tượng <*label*> thứ hai dẫn đến đối tượng <*label*> thứ hai có chữ màu đỏ. Đối tượng <*label*> thứ ba được chọn bởi hai bộ chọn *label* và bộ chọn *div>label*, đồng thời kế thừa bảng định dạng có bộ chọn *#d*. Bộ chọn *#d* có tính cụ thể cao nhất. Tuy vậy, quy tắc thứ nhất đã không cho đối tượng <*label*> thứ ba được áp dụng màu xanh da trời từ bảng định dạng của bộ chọn *#d*. Thay vào đó, quy tắc thứ hai đã

xác định cho đối tượng *<label>* thứ ba có chữ màu tím từ bảng định dạng của bộ chọn *div>label*. Kết quả là đối tượng *<label>* thứ ba có chữ màu tím và in đậm.

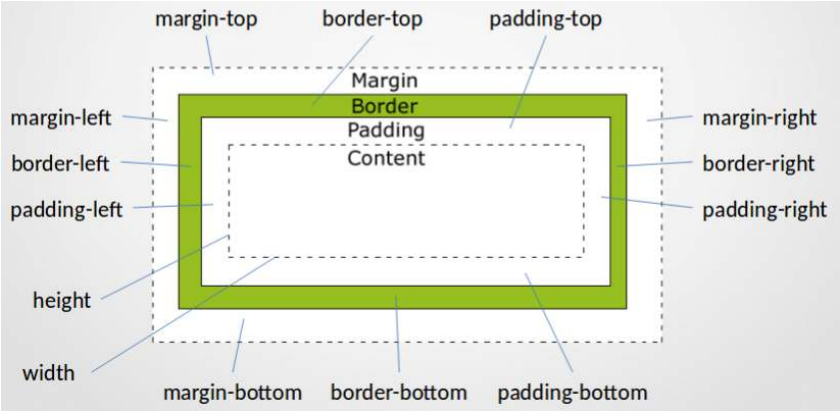
```
1. /* Filename: precedence.css */
2. label {color:yellow;}
```

```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.6</title>
3.   <meta charset="utf-8">
4.     <style type="text/css">
5.         label.note {color:red;}
6.         label {color:green;}
7.         div>label {color:purple;}
8.         #d {color:blue; font-weight:bold;}
9.     </style>
10.    <link rel="stylesheet" href="precedence.css">
11.</head><body>
12.  <label>Nhãn 1</label>
13.  <label class="note">Nhãn 2</label>
14.  <div id="d"><label>Nhãn 3</label></div>
15.</body></html>
```


CSS cho phép thực thi quy tắc thứ hai, xác định tính cụ thể, với ngoại lệ *!important* đặt sau các giá trị thuộc tính. Thuộc tính có ngoại lệ *!important* sẽ có tính cụ thể cao hơn thuộc tính không có *!important*. Ví dụ, nếu ở trang web trước, ngoại lệ *!important* được đặt vào thuộc tính *color* trong bảng định dạng thứ hai dạng "*label {color: green !important;}*" thì cả ba đối tượng *<label>* đều được hiển thị với chữ màu xanh lá cây. Ngược lại, nếu ngoại lệ *!important* được đặt vào thuộc tính *color* trong bảng định dạng thứ tư dạng "*#d {color:blue !important; font-weight:bold;}*" thì các đối tượng *<label>* vẫn được hiển thị với các màu như trước do bảng định dạng thứ tư này chỉ là bảng định dạng kế thừa trên đối tượng *<label>* thứ ba. Ngoại lệ *!important* được khuyến cáo hạn chế sử dụng vì nó làm xáo trộn các quy tắc xác định thứ tự ưu tiên dẫn đến khó kiểm soát chương trình.

3.7. MÔ HÌNH TRÌNH DIỄN ĐỐI TƯỢNG TÀI LIỆU

Các đối tượng tài liệu được trình diễn trên giao diện của trình duyệt theo **mô hình hộp** (box model) như được minh họa trong Hình 3.1. Theo mô hình này, nội dung của đối tượng tài liệu được hiển thị trong một hình chữ nhật được gọi là hộp nội dung (content box). Bao quanh hộp nội dung là đường viền (border). Đường viền và hộp nội dung được phân cách nhau bởi phần đệm (padding). Đệm là khoảng trống trong suốt có tác dụng không cho nội dung và đường viền dính liền nhau khi hiển thị. Bao quanh bên ngoài viền là khoảng trống trong suốt khác được gọi là lề (margin). Lề có tác dụng phân cách các đối tượng tài liệu khi chúng được hiển thị cạnh nhau.



Hình 3.1. Mô hình hộp trình diễn đối tượng tài liệu.

Kích thước hộp nội dung thường được trình duyệt tính toán tự động để toàn bộ nội dung được nhìn thấy. Tuy nhiên, lập trình viên có thể ấn định kích thước hộp này bằng các thuộc tính *width* (chiều rộng), *height* (chiều cao), *max-width* (chiều rộng tối đa), *min-width* (chiều rộng tối thiểu), *max-height* (chiều cao tối đa), *min-height* (chiều cao tối thiểu). Giá trị được gán cho các thuộc tính kể trên là độ dài, sử dụng các đơn vị *px* (pixel), *pt* (point), hoặc % (phần trăm theo đối tượng cha), ví dụ "*width:100px;*" hay "*max-width:50%;*".

Khác với hộp nội dung, đường viền có nhiều thuộc tính hơn. Các thuộc tính này cho phép chỉ định kiểu dáng, màu sắc và độ dày cho toàn bộ viền (*border*) hay viền ở từng phía (*boder[-side]* với *[-side]* là *-top, -left, -right* hay *-bottom*). Các thuộc tính cho viền được tóm tắt như sau:

Mục đích định dạng	**Thuộc tính: giá trị có thể nhận**
Kiểu dáng viền	*border[-side]-style: solid|dashed|dotted|…*
Màu sắc viền	*border[-side]-color: color-value*
Độ dày viền	*border[-side]-width: thin/medium/thick/#px*
Cả ba ở trên	*border[-side]: style width color*

Đệm và lề chỉ được xác định bằng độ dày. Đối tượng tài liệu có các thuộc tính cho toàn bộ đệm/lề hay các đệm/lề ở từng phía như được chỉ ra trên Hình 3.1.

Nếu không được chỉ định cụ thể, viền, đệm và lề của mỗi đối tượng tài liệu được xác định bởi bảng định dạng mặc định của trình duyệt.

Trong trang web ví dụ sau đây, một đối tượng tiêu đề được khai báo, viền, đệm và lề của đối tượng được xác định cụ thể.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.7</title>
3.   <meta charset="utf-8">
4.   <style type="text/css">
5.     h1 {
6.       border:solid 2px red;
7.       padding:2px 15px; margin:10px;
```

```
8.      }
9.    </style>
10.</head><body>
11.   <h1>Tiêu đề 1</h1>
12.</body></html>
```

3.8. HIỂN THỊ THEO DÒNG VÀ THEO KHỐI

Đối tượng tài liệu có thể được hiển thị theo dòng (inline) hoặc theo khối (block). Thuộc tính *display* của đối tượng tài liệu được sử dụng để xác định kiểu hiển thị. Một đối tượng hiển thị theo khối sẽ chiếm toàn bộ dòng mà nó có mặt, không thể đứng cạnh các đối tượng khác trên cùng dòng, cho dù khoảng trống phía sau của dòng còn đủ để hiển thị các đối tượng khác. Cũng có thể hình dung là trình duyệt sẽ tự động thêm các đối tượng ngắt dòng (*
) vào trước và sau đối tượng hiển thị theo khối. Ngược lại, các đối tượng hiển thị theo dòng có thể được trình diễn trên cùng dòng, đối tượng này cạnh đối tượng kia. Đặc biệt, nếu khoảng trống còn lại của dòng không còn đủ, hoặc nội dung bên trong có chứa đối tượng ngắt dòng (
*), hộp của đối tượng hiển thị theo dòng sẽ được gấp thành nhiều đoạn và hiển thị trên nhiều dòng liên tiếp. Đối tượng hiển thị theo dòng không cho đặt giá trị cho các thuộc tính *width*, *height*, *margin-top*, và *margin-bottom*. Các giá trị này được tính tự động. Lưu ý, hộp của đối tượng hiển thị theo khối không bị gấp ngay cả khi bề rộng của nó lớn hơn bề rộng của vùng hiển thị.

Cũng có thể đặt kiểu hiển thị cho đối tượng là dòng-khối (inline-block). Hiển thị theo dòng-khối tương tự hiển thị theo khối, ngoại trừ cho phép các đối tượng đứng trên cùng dòng.

Trong trang web ví dụ sau đây, chín đối tượng nhãn được khai báo, trong đó ba đối tượng đầu tiên được hiển thị theo dòng (mặc định của nhãn), ba đối tượng tiếp theo được hiển thị theo khối và ba đối tượng cuối cùng được hiển thị theo dòng-khối. Nội dung các nhãn thứ hai, năm và tám có chứa đối tượng ngắt dòng. Do đó, hộp của nhãn thứ hai được gấp đôi, một nửa được hiển thị ở đầu dòng tiếp theo, trong khi hộp của các nhãn thứ năm và tám có bề rộng xác định (150 pixels) và nội dung bên trong nó được hiển thị theo dòng nội trong hộp. Tương tự, nội dung các nhãn thứ ba, sáu và chín rất dài dẫn đến hộp của nhãn thứ ba cũng được gấp trên nhiều dòng trong khi hộp của các nhãn thứ sáu và chín thì không. Các hộp của nhãn thứ bảy, tám và chín được hiển thị cùng dòng vì các nhãn này có kiểu hiển thị là dòng-khối.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.8</title>
3.   <meta charset="utf-8">
4.   <style type="text/css">
5.     label {border:solid 2px red;  padding:2px 15px;
6.              margin:10px; width:150px;}
7.     label.b {display:block;}
8.     label.ib {display:inline-block;}
9.   </style>
```

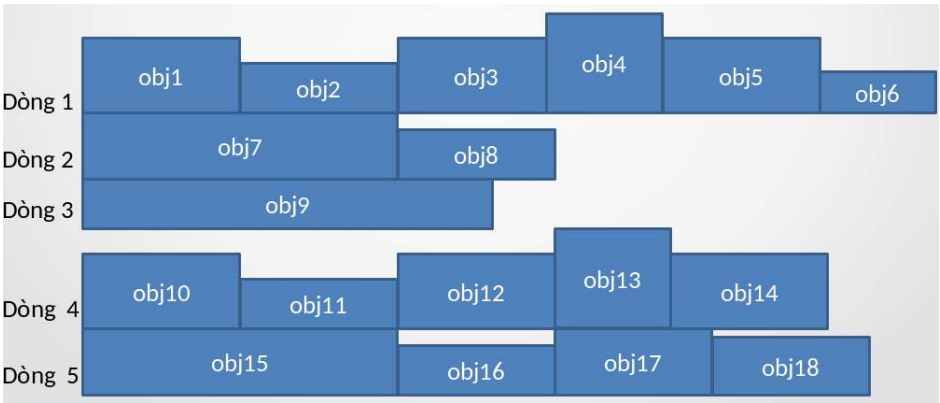
```
10. </head><body>
11.   <label>Nhãn 1</label>
12.   <label>Nhãn <br> 2</label>
13.   <label>Nhãn nhãn nhãn nhãn nhãn nhãn nhãn nhãn nhãn nhãn 3</label>
14.   <label class="b">Nhãn 4</label>
15.   <label class="b">Nhãn <br> 5</label>
16.   <label class="b">Nhãn nhãn nhãn nhãn nhãn nhãn nhãn nhãn 6</label>
17.   <label class="ib">Nhãn 7</label>
18.   <label class="ib">Nhãn <br> 8</label>
19.   <label class="ib">Nhãn nhãn nhãn nhãn nhãn nhãn nhãn nhãn 9</label>
20. </body></html>
```

3.9. VỊ TRÍ TRÌNH DIỄN ĐỐI TƯỢNG TÀI LIỆU

Khi đã biết mỗi đối tượng tài liệu được trình diễn như một cái hộp (Mục 3.7), việc cuối cùng cần thực hiện để hoàn tất trình diễn là xác định vị trí đặt chiếc hộp đó trên giao diện của trình duyệt. Thuộc tính *position* của đối tượng tài liệu được sử dụng cho mục đích này. Có bốn phương thức xác định vị trí trình diễn các đối tượng tài liệu là tĩnh (static), tương đối (relative), tuyệt đối (absolute) và cố định (fixed). Ngoài ra, có thể sử dụng ngoại lệ trôi (*float*) để thay đổi vị trí trình diễn. Các phương thức trình diễn và ngoại lệ sẽ được trình bày chi tiết ngay sau đây.

3.9.1. Vị trí tĩnh

Xác định vị trí tĩnh (*position:static*) là phương thức mặc định được áp dụng cho các đối tượng tài liệu. Với phương thức này, các đối tượng theo thứ tự khai báo lần lượt được hiển thị theo **luồng thông thường** (normal flow), tức là theo các dòng, từ trái qua phải rồi từ trên xuống dưới. Lưu ý, mỗi đối tượng có kiểu hiển thị theo khối sẽ chiếm một dòng mà không chung dòng với bất kỳ đối tượng nào khác. Chiều cao của một dòng bằng chiều cao hộp của đối tượng cao nhất cùng dòng. Các đối tượng trên cùng dòng được căn lề dưới theo mặc định. Hình 3.2 minh họa phương thức xác định vị trí theo luồng thông thường, trong đó đối tượng thứ chín (*obj9*) có kiểu hiển thị theo khối, các đối tượng còn lại có kiểu hiển thị theo dòng.



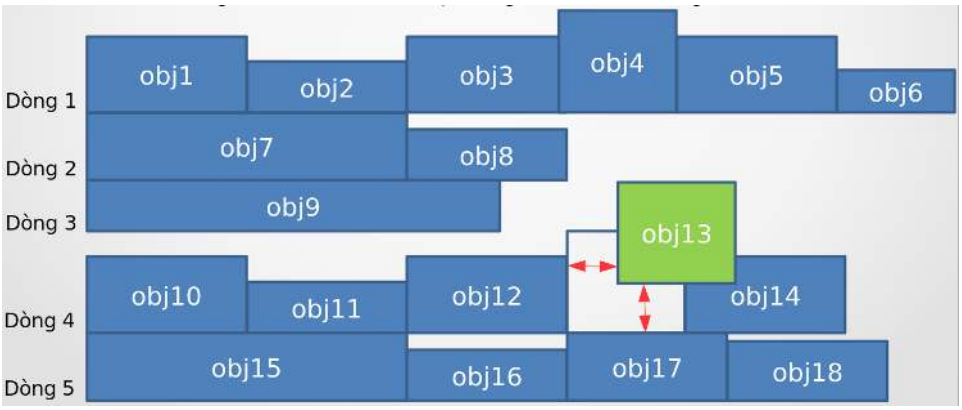
Hình 3.2. Các đối tượng được hiển thị theo luồng thông thường.

Trang web sau đây là cụ thể hóa ví dụ được minh họa ở Hình 3.2. Các đối tượng nhãn có kiểu hiển thị theo dòng theo mặc định. Đối tượng thứ chín được đặt kiểu hiển thị theo khối. Tất cả các đối tượng được hiển thị theo luồng bình thường.

```
1.  <!DOCTYPE html><html><head>
2.    <title>L.3.9.1</title>
3.    <meta charset="utf-8">
4.    <style type="text/css">
5.      label {border: solid 2px red;}
6.      label.b {display:block;}
7.    </style>
8.  </head><body>
9.    <label>Nhãn 1</label><label>Nhãn 2</label><label>Nhãn 3</label>
10.   <label>Nhãn 4</label><label>Nhãn 5</label><label>Nhãn 6</label>
11.   <label>Nhãn 7</label><label>Nhãn 8</label>
12.   <label class="b">Nhãn 9</label>
13.   <label>Nhãn 10</label><label>Nhãn 11</label><label>Nhãn 12</label>
14.   <label>Nhãn 13</label><label>Nhãn 14</label><label>Nhãn 15</label>
15.   <label>Nhãn 16</label><label>Nhãn 17</label><label>Nhãn 18</label>
16. </body></html>
```


3.9.2. Vị trí tương đối

Vị trí trình diễn của đối tượng có vị trí tương đối (*position:relative*) được xác định thông qua các độ rời (offset) trái/phải và trên/dưới so với vị trí tĩnh của nó. Nói cách khác, vị trí trình diễn của đối tượng là vị trí tương đối so với vị trí tĩnh trong luồng thông thường. Lưu ý vị trí tĩnh trong luồng thông thường vẫn bị chiếm chỗ khi đối tượng có vị trí tương đối đã rời đi. Đối tượng có vị trí tương đối có thể được hiển thị chồng lên/che mất các đối tượng khác. Trong ví dụ minh họa ở Hình 3.3, đối tượng thứ mười ba (*obj13*) có vị trí tương đối được xác định thông qua các độ rời trái (left) và dưới (bottom), các đối tượng còn lại có vị trí tĩnh.



Hình 3.3. Một đối tượng có vị trí tương đối.

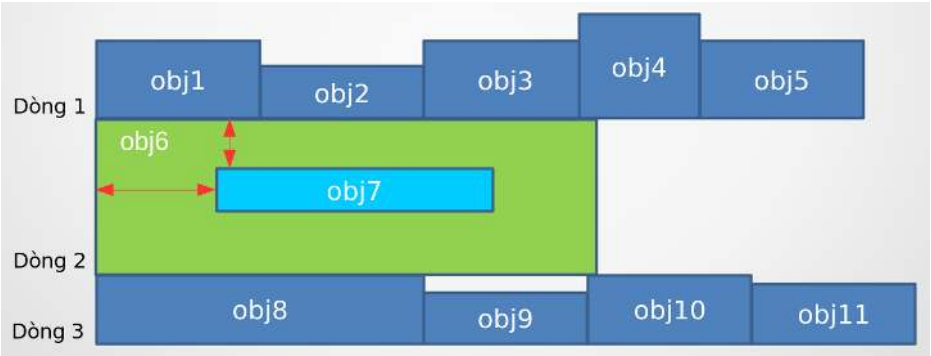
Trang web sau đây là cụ thể hóa ví dụ được minh họa ở Hình 3.3. Tất cả các đối tượng được hiển thị theo luồng bình thường ngoại trừ đối tượng thứ mười ba có vị trí tương đối.

```
1.  <!DOCTYPE html><html><head>
2.    <title>L.3.9.2</title>
```

```
3.    <meta charset="utf-8">
4.    <style type="text/css">
5.      label {border:solid 2px red;}
6.      label.b {display:block;}
7.      label.r {position:relative; left:20px; bottom:20px;}
8.    </style>
9.  </head><body>
10.   <label>Nhãn 1</label><label>Nhãn 2</label><label>Nhãn 3</label>
11.   <label>Nhãn 4</label><label>Nhãn 5</label><label>Nhãn 6</label>
12.   <label>Nhãn 7</label><label>Nhãn 8</label>
13.   <label class="b">Nhãn 9</label>
14.   <label>Nhãn 10</label><label>Nhãn 11</label><label>Nhãn 12</label>
15.   <label class="r">Nhãn 13</label>
16.   <label>Nhãn 14</label><label>Nhãn 15</label>
17.   <label>Nhãn 16</label><label>Nhãn 17</label><label>Nhãn 18</label>
18. </body></html>
```

3.9.3. Vị trí tuyệt đối

Vị trí trình diễn của đối tượng có vị trí tuyệt đối (*position:absolute*) được xác định thông qua các độ rời trái/phải và trên/dưới so với vị trí của đối tượng tổ tiên gần nhất không theo luồng thông thường. Nói cách khác, vị trí tuyệt đối là vị trí tương đối so với vị trí của tổ tiên gần nhất có vị trí khác tĩnh. Vị trí tĩnh trong luồng thông thường không bị chiếm chỗ khi đối tượng có vị trí tuyệt đối đã rời đi. Đối tượng có vị trí tuyệt đối có thể được hiển thị chồng lên/che mất các đối tượng khác. Trong ví dụ minh họa ở Hình 3.4, đối tượng thứ sáu (*obj6*) chứa đối tượng thứ bảy (*obj7*), và đối tượng thứ sáu có vị trí tương đối trong khi đối tượng thứ bảy có vị trí tuyệt đối với các độ rời trái (left) và trên (top) so với vị trí của đối tượng thứ sáu.



Hình 3.4. Một đối tượng có vị trí tuyệt đối.

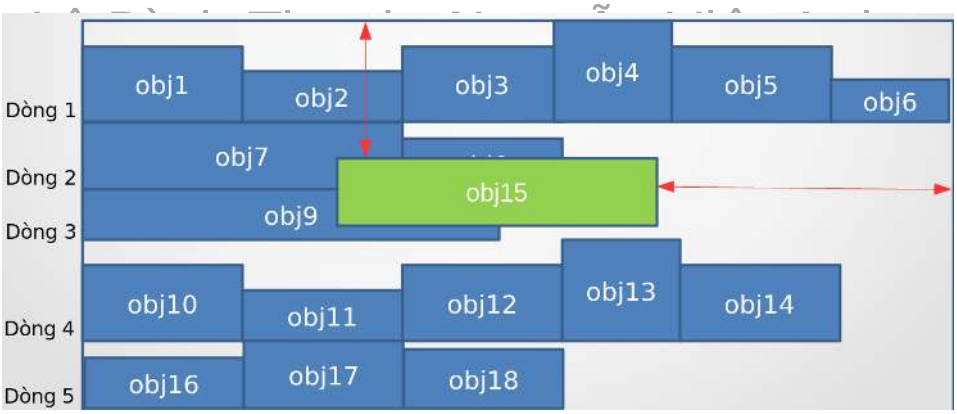
Trang web sau đây là cụ thể hóa ví dụ được minh họa ở Hình 3.4. Tất cả các đối tượng được hiển thị theo luồng bình thường ngoại trừ đối tượng thứ bảy có vị trí tuyệt đối và đối tượng thứ sau có vị trí tương đối.

```
1.  <!DOCTYPE html><html><head>
2.    <title>L.3.9.3</title>
3.    <meta charset="utf-8">
4.    <style type="text/css">
5.      div, label {border:solid 2px red;}
6.      div {position:relative; height:150px;}
```

```
7.      label.abs {position:absolute; left:30px; top:50px;}
8.    </style>
9.  </head><body>
10.   <label>Nhãn 1</label><label>Nhãn 2</label><label>Nhãn 3</label>
11.   <label>Nhãn 4</label><label>Nhãn 5</label>
12.   <div>Div 6  <label class="abs">Nhãn 7</label></div>
13.   <label>Nhãn 8</label><label>Nhãn 9</label>
14.   <label>Nhãn 10</label><label>Nhãn 11</label>
15. </body></html>
```

3.9.4. Vị trí cố định

Vị trí trình diễn của đối tượng có vị trí cố định (*position:fixed*) được xác định thông qua các độ rời trái/phải và trên/dưới so với cửa sổ hiển thị. Đối tượng có vị trí cố định có thể được hiển thị chồng lên/che mất các đối tượng khác và không bị trôi theo thanh cuộn khi vị trí thanh cuộn được thay đổi. Trong ví dụ minh họa ở Hình 3.5, đối tượng thứ mười lăm (*obj15*) có vị trí cố định với các độ rời phải (right) và trên (top) so với vị trí cửa sổ.



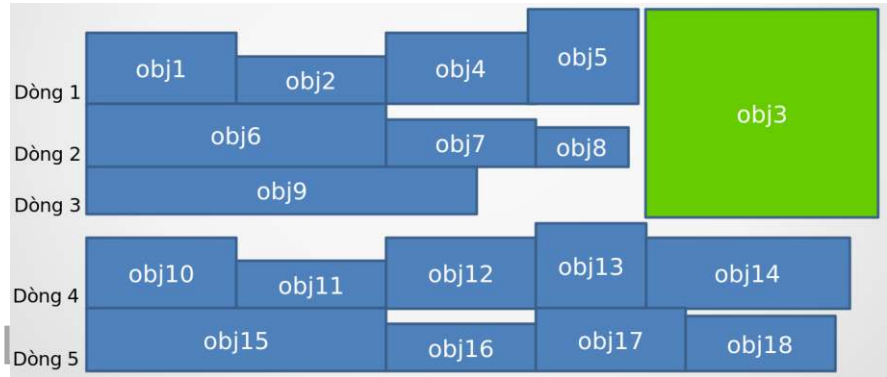
Hình 3.5. Một đối tượng có vị trí cố định.

Trang web sau đây là cụ thể hóa ví dụ được minh họa ở Hình 3.5. Tất cả các đối tượng được hiển thị theo luồng bình thường ngoại trừ đối tượng thứ mười lăm có vị trí cố định.

```
1.  <!DOCTYPE html><html><head>
2.    <title>L.3.9.4</title>
3.    <style type="text/css">
4.      label {border:solid 2px red;}
5.      label.b {display:block;}
6.      label.f {position:fixed; top:20px; right:250px;}
7.    </style>
8.  </head><body>
9.    <label>Nhãn 1</label><label>Nhãn 2</label><label>Nhãn 3</label>
10.   <label>Nhãn 4</label><label>Nhãn 5</label><label>Nhãn 6</label>
11.   <label>Nhãn 7</label><label>Nhãn 8</label>
12.   <label class="b">Nhãn 9</label>
13.   <label>Nhãn 10</label><label>Nhãn 11</label><label>Nhãn 12</label>
14.   <label>Nhãn 13</label><label>Nhãn 14</label>
15.   <label class="f">Nhãn 15</label>
16.   <label>Nhãn 16</label><label>Nhãn 17</label><label>Nhãn 18</label>
17. </body></html>
```

3.9.5. Trôi

Ngoài *position*, các đối tượng tài liệu còn có thuộc tính *float* (trôi) để xác định vị trí trình diễn. Đối tượng được đặt trôi được dạt về một bên, trái hoặc phải, của vùng hiển thị. Các đối tượng sau đối tượng đặt trôi chảy quanh đối tượng đặt trôi. Trong ví dụ minh họa ở Hình 3.6, đối tượng thứ ba (*obj3*) được trôi phải, các đối tượng sau nó (*obj4* trở đi) chảy quanh đối tượng thứ ba. Ứng dụng điển hình nhất của đặt trôi là bao văn bản quanh hình ảnh.



Hình 3.6. Một đối tượng được đặt trôi.

Trang web sau đây là cụ thể hóa ví dụ được minh họa ở Hình 3.6. Lưu ý, trong trang web này, đối tượng *
* với thuộc tính *clear* được sử dụng. Thuộc tính *clear* có tác dụng ngắt trôi, nghĩa là làm cho các đối tượng sau không chảy quanh đối tượng được đặt trôi nữa. Trong trang web này, * * và các *<label>* sau * * sẽ được hiển thị phía dưới đối tượng nhãn thứ ba. Nếu xóa thuộc tính *clear* của *
*, * * và các *<label>* phía sau sẽ chảy quanh đối tượng nhãn thứ ba như các đối tượng nhãn trước đó.

```
1.  <!DOCTYPE html><html><head>
2.    <title>L.3.9.5</title>
3.    <meta charset="utf-8">
4.    <style type="text/css">
5.      label, div {border:solid 1px red;}
6.      label.fl {float:right; width:300px; height:200px;}
7.      br.cl {clear:both;}
8.      </style>
9.  </head><body>
10.   <label>Nhãn 1</label><label>Nhãn 2</label>
11.   <label class="fl">Nhãn 3</label>
12.   <label>Nhãn 4</label><label>Nhãn 5</label><label>Nhãn 6</label>
13.   <label>Nhãn 7</label><label>Nhãn 8</label><label>Nhãn 9</label>
14.   <label>Nhãn 10</label><label>Nhãn 11</label><label>Nhãn 12</label>
15.   <label>Nhãn 13</label><label>Nhãn 14</label><label>Nhãn 15</label>
16.   <label>Nhãn 16</label><label>Nhãn 17</label><label>Nhãn 18</label>
17.   <br class="cl">
18.   <div>div</div>
19.   <label>Nhãn 19</label><label>Nhãn 20</label><label>Nhãn 21</label>
20. </body></html>
```

Một điểm nữa cần lưu ý khi đặt trôi là chiều cao của các đối tượng. Nếu chiều cao của đối tượng đặt trôi lớn hơn chiều cao của đối tượng chứa nó, hiển thị của đối tượng đặt trôi sẽ tràn ra bên ngoài đối tượng chứa. Để khắc phục điểm này, đối tượng chứa cần sử dụng thuộc tính *overflow* với giá trị là "auto".

3.9.6. Cao độ

Các đối tượng có vị trí tương đối, tuyệt đối, hoặc cố định có thể che/nằm trước các đối tượng khác khi các vùng hiển thị của chúng giao nhau. Thuộc tính *z-index* (cao độ) cho phép xác định đối tượng nào che/nằm trước đối tượng nào. *z-index* nhận các giá trị nguyên. Đối tượng có *z-index* cao hơn sẽ che/nằm trước đối tượng có *z-index* nhỏ hơn khi các vùng hiển thị của chúng giao nhau.

3.10. CSS CÓ ĐIỀU KIỆN

Có thể gộp các bảng định dạng thành nhóm và đặt điều kiện áp dụng lên nhóm. Điều kiện là một biểu thức logic mà chỉ khi biểu thức đúng (true) thì các bảng định dạng trong nhóm mới được áp dụng. Mục đích sử dụng CSS có điều kiện là để thiết kế giao diện có thể thích ứng với thiết bị. Nói cách khác, CSS có điều kiện cho phép áp dụng các bảng định dạng khác nhau tùy thuộc vào đặc tính của thiết bị và trình duyệt.

Có nhiều cách thức khác nhau để khai báo CSS có điều kiện. Giáo trình này chỉ giới thiệu một số cách thức phổ biến nhất, được hầu hết các trình duyệt hiện đại hỗ trợ. Những cách thức ít phổ biến hơn hoặc được ít trình duyệt hỗ trợ không được trình bày.

Cách thức đầu tiên thông dụng là chỉ định *điều kiện hỗ trợ* (supports condition) của trình duyệt đối với các thuộc tính CSS. Rõ ràng, nếu trình duyệt không hỗ trợ một thuộc tính CSS nào đó thì không thể/nên áp dụng bảng định dạng. Cú pháp chỉ định điều kiện hỗ trợ như sau:

@supports <điều_kiện_hỗ_trợ> {

Các bảng định dạng

}

trong đó, *<điều_kiện_hỗ_trợ>* bao gồm một hoặc nhiều cặp *thuộc_tính:giá_trị* được kết hợp bởi các toán tử *and*, *or* và *not*. Nếu *<điều_kiện_hỗ_trợ>* được thỏa mãn, các bảng định dạng được khai báo trong nhóm mới được áp dụng. Ví dụ, hai bảng định dạng được khai báo sau đây chỉ được áp dụng nếu trình duyệt hỗ trợ kiểu hiển thị *flexbox* và không hỗ trợ kiểu hiển thị *inline-grid*.

```
1. @supports (display:flexbox) and (not (display:inline-grid)) {
2.   div.box {display:flexbox;}
3.   span {display:flexbox;}
4. }
```

Cách thức thứ hai thông dụng để khai báo CSS có điều kiện là sử dụng *truy vấn phương tiện* (media query). Với truy vấn phương tiện, CSS có điều kiện được khai báo như sau:

@media <(danh_sách)truy_vấn_phương_tiện> {

Các bảng định dạng

}

trong đó, biểu thức điều kiện *<(danh_sách)truy_vấn_phương_tiện>* bao gồm một hoặc nhiều truy vấn phương tiện. Cũng có thể sử dụng truy vấn phương tiện để đặt điều kiện bao hàm CSS ngoài như sau:

@import url("file.css") <(danh_sách)truy_vấn_phương_tiện>;

Khi đó tệp *file.css* chỉ được bao hàm nếu biểu thức điều kiện *<(danh_sách)truy_vấn_phương_tiện>* đúng. Ngoài ra, có thể đặt truy vấn phương tiện vào thẻ *<link>* trong khai báo HTML như sau:

<link rel="stylesheet" type="text/css" href="file.css"

media="<(danh_sách)truy_vấn_phương_tiện>">

Khi đó, điều tương tự xảy ra, tức tệp *file.css* chỉ được bao hàm nếu biểu thức điều kiện *<(danh_sách)truy_vấn_phương_tiện>* đúng.

Cú pháp của truy vấn phương tiện như sau:

[*not*|*only*]? *<kiểu_phương_tiện>* | (*<đặc_điểm_của_phương_tiện>)* [and (*<đặc_điểm_của_phương_tiện>*)]*

Một truy vấn phương tiện bao gồm không hoặc một kiểu phương tiện và không hoặc nhiều đặc điểm của phương tiện. Kiểu phương tiện có thể nhận các giá trị: *all* (mọi thiết bị), *print* (máy in), *screen* (máy tính, tablet, smart-phone) và *speech* (screenreader). Nếu kiểu phương tiện không được chỉ định, giá trị *all* sẽ được sử dụng. Đặc điểm phương tiện được dùng để chỉ kích thước màn hình, kích thước vùng hiển thị, độ phân giải của màn hình, số bít màu của thiết bị, ..., dưới dạng cặp *thuộc_tính:giá_trị*. Thiết bị phải có kiểu phương tiện cùng các đặc điểm phương tiện khớp với truy vấn phương tiện thì truy vấn phương tiện mới trả về giá trị là đúng (true). Ngược lại, kiểu phương tiện hoặc ít nhất một đặc điểm phương tiện không được thỏa mãn, truy vấn phương tiện trả về giá trị sai (false). Trường hợp có từ khóa *'not'* ở trước, giá trị của truy vấn phương tiện sẽ được đảo ngược. Từ khóa *'only'* không tham gia giá trị của truy vấn mà được sử dụng cho mục đích khác. Từ khóa này nói rằng nếu trình duyệt (cũ) không hỗ trợ (không hiểu) truy vấn phương tiện thì hãy bỏ qua các lớp CSS đi kèm, ngược lại trình duyệt hỗ trợ truy vấn phương tiện thì loại bỏ từ khóa *'only'* này ra khỏi truy vấn trước khi xử lý truy vấn.

Có thể kết hợp các truy vấn phương tiện thành một danh sách với các truy vấn được phân cách nhau bởi dấu phẩy (,). Danh sách có giá trị đúng nếu ít nhất một truy vấn đúng. Danh sách có giá trị sai nếu tất cả các truy vấn đều sai.

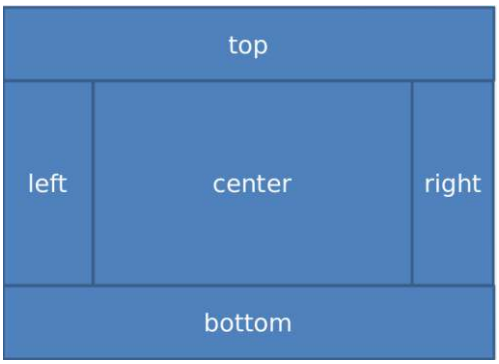
Ví dụ về sử dụng truy vấn phương tiện, ba bảng định dạng sau đây sẽ được sử dụng nếu thiết bị được sử dụng là máy tính, tablet, smart-phone và chiều rộng tối đa của vùng hiển thị là 500px.

```
1. @media screen and (max-width: 500px) {
2.   .gridmenu {  width:100%; }
3.   .gridmain {  width:100%; }
4.   .gridright { width:100%; }
5. }
```

3.11. MỘT VÀI ĐỊNH DẠNG PHỔ BIẾN

3.11.1. Dàn trang

Dàn trang (layout) là công việc đầu tiên và quan trọng trong việc trình diễn trang web. Dàn trang chia vùng hiển thị thành các vùng con để mỗi vùng con đóng một vai trò và trình diễn một nội dung khác các vùng con khác. Ví dụ điển hình của dàn trang được thể hiện trong Hình 3.7. Trong ví dụ này, vùng hiển thị, có thể là cửa sổ của trình duyệt, được chia thành các vùng trên (top), giữa (middle) và dưới (bottom) theo chiều đứng; vùng giữa được chia tiếp thành các vùng trái (left), trung tâm (center) và phải (right) theo chiều ngang.



Hình 3.7. Một ví dụ và dàn trang.

Để chia một vùng thành các vùng con theo chiều đứng, đơn giản tạo mỗi vùng con bằng một * * với kiểu hiển thị mặc định theo khối và độ rộng mặc định là 100%. Tương tự, để chia một vùng thành các vùng con theo chiều ngang, hãy tạo mỗi vùng con bằng một * * với kiểu hiển thị dòng-khối và độ rộng xác định. Ví dụ dàn trang ở Hình 3.7 có thể được cài đặt như sau.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.3.11.1</title>
3.   <meta charset="utf-8">
4.   <style type="text/css">
5.     div {border:solid 1px red;}
```

```
6.      #left, #right, #center {
7.        display:inline-block;
8.        vertical-align:top;
9.        width:200px;
10.     }
11.   </style>
12. </head><body>
13.   <div id="container">
14.     <div id="top">a</div>
15.     <div id="middle">
16.       <div id="left">b<br>b</div>
17.       <div id="center">c</div>
18.       <div id="right">d</div>
19.     </div>
20.     <div id="bottom">e</div>
21.   </div>
22. </body></html>
```

Cũng có thể sử dụng đặt trôi để dàn trang theo chiều ngang. Tuy nhiên, sử dụng kiểu hiển thị dòng-khối là dễ dàng hơn cả.

3.11.2. Giá trị màu

Màu được sử dụng để đặt cho chữ, đường viền hay nền của các đối tượng. Với những màu phổ biến, có thể xác định giá trị màu theo tên, ví dụ *red*, *green*, *blue*. Với màu bất kỳ, giá trị màu được xác định theo RGB, ví dụ *rgb*(255, 0, 0) hoặc #ff0000.

3.11.3. Định dạng văn bản, phông chữ

Các thuộc tính để định dạng văn bản, phông chữ được liệt kê dưới đây kèm theo giá trị của chúng. Mỗi thuộc tính hoặc giá trị đều đã tự mô tả nó.

color:#00ff00;

text-align:center | left | right | justify;

vertical-align:bottom | top | middle;

text-decoration:none | underline | line-through | overline/blink;

text-transform:uppercase | lowercase | capitalize;

text-indent:50px;

word-wrap: normal | break-word;

text-overflow: clip | ellipsis | string;

font-family:"Times New Roman", Times, serif;

font-size:40px;

font-style:normal | italic | bold;

font-weight: normal | bold | bolder | lighter | number | initial | inherit;

3.11.4. Định dạng nền

Các thuộc tính để định dạng nền được liệt kê dưới đây kèm theo giá trị của chúng. Mỗi thuộc tính hoặc giá trị đều đã tự mô tả nó.

background-color: #6495ed;

background-image:url('paper.gif');

background-repeat:repeat-x;

background-attachment:fixed;

background-position:right top;

background:#ffffff url('img_tree.png') no-repeat right top;

3.11.5. Định dạng viền

Các thuộc tính để định dạng viền được liệt kê dưới đây kèm theo giá trị của chúng. Mỗi thuộc tính hoặc giá trị đều đã tự mô tả nó.

border:2px solid;

border-radius:25px;

border-image:url(border.png) 30 30 round;

3.11.6. Biến đổi 2D, 3D

Các thuộc tính để biến đổi 2D, 3D được liệt kê dưới đây kèm theo giá trị của chúng. Mỗi thuộc tính hoặc giá trị đều đã tự mô tả nó.

transform: rotate(30deg);

transform: translate(50px,100px);

transform: scale(2,4);

transform: skew(30deg,20deg);

transform: rotateX(120deg);

transform: rotateY(130deg);

3.12. CẬP NHẬT KIẾN THỨC VỀ CSS

Đặc tả CSS liên tục được W3C nâng cấp, sửa đổi (phiên bản hiện tại là 3). Một số kiểu trình diễn hay bộ chọn đang có sẽ trở nên lỗi thời trong khi một số kiểu trình diễn hay bộ chọn mới sẽ được bổ sung. Vì vậy, việc nắm vững vai trò định kiểu trình diễn đối tượng tài liệu của CSS cùng với việc cập nhật kiến thức về các phiên bản CSS là yêu cầu bắt buộc đối với các lập trình viên web.

Viết mã CSS là công việc được lặp đi lặp lại và tốn thời gian, đòi hỏi sự tỉ mỉ, đôi khi là buồn chán. Hiểu được những khó khăn đó, một số thư viện tiền xử lý CSS như Sass[10] và Less[11] đã ra đời. Tiền xử lý CSS về bản chất là sử dụng các ngôn ngữ kịch bản để mở rộng CSS, cho phép thêm các biến, hàm và nhiều kỹ thuật để tạo ra CSS. Bộ tiền xử lý CSS sẽ biên dịch và chạy kịch bản để sinh ra CSS. Việc sử dụng kịch bản, tức tiền xử lý, giúp tiết kiệm thời gian, không phải lặp đi lặp lại, dễ bảo trì và mở rộng trong việc tạo CSS.

Bài tập

1. Tạo các trang HTML, mỗi trang nhập một mã nguồn được viết trong các ví dụ được trình bày ở chương này, nhập xong mỗi ví dụ thì tải trang bằng trình duyệt để thấy thể hiện của trang trên giao diện.
2. Từ ví dụ dàn trang trong Mục 3.11.1, hãy bổ sung mã nguồn CSS để trang web có khả năng thích ứng, tức là thay đổi dàn trang theo kích thước cửa sổ hiển thị.

Đọc thêm

1. Lea Verou, "CSS Secrets: Better Solutions to Everyday Web Design Problems, 1st Edition", O'Reilly Media, 2015.
2. Andy Budd, Emil Björklund, "CSS Mastery, 3rd Edition", Apress, 2013.
3. Anirudh Prabhu, ‚Beginning CSS Preprocessors: With SASS, Compass.js and Less.js, 1st Edition`, Apress, 2015.

[10] http://sass-lang.com/
[11] http://lesscss.org/

Chương 4
QUẢN LÝ TRANG WEB BẰNG JAVASCRIPT

4.1. CƠ BẢN VỀ JAVASCRIPT

JavaScript là ngôn ngữ thứ ba được sử dụng để tạo ra nội dung web. Cùng với HTML và CSS, JavaScript là một trong những công nghệ lõi của web. Nếu như HTML được sử dụng để khai báo và CSS được sử dụng để xác định kiểu trình diễn, thì JavaScript được sử dụng để quản lý các đối tượng tài liệu. HTML cung cấp các thẻ khai báo các đối tượng tài liệu nhưng không cung cấp khả năng quản lý chúng. Ví dụ, thẻ *<button>* tạo một nút bấm nhưng không xử lý sự kiện khi nút được bấm. Sử dụng JavaScript có thể quản lý toàn bộ trang web nói chung, thêm mới, hủy bỏ, thay đổi thuộc tính, triệu gọi phương thức của các đối tượng tài liệu bất cứ khi nào. Mục này trình bày những nội dung cơ bản về ngôn ngữ lập trình JavaScript. Sử dụng JavaScript để quản lý trang web sẽ được trình bày trong các mục tiếp theo. Việc chèn mã JavaScript vào trang web đã được trình bày tại Mục 2.9 trong Chương 2.

JavaScript là ngôn ngữ lập trình **dựa trên nguyên mẫu** (prototype-based), hỗ trợ các phương pháp lập trình mệnh lệnh, thủ tục và hướng đối tượng. Nó có các API cho làm việc với văn bản, mảng, đối tượng, biểu thức chính quy nhưng không hỗ trợ vào/ra như lưu trữ, kết nối mạng. JavaScript có nhiều điểm tương đồng với Java và C, từ các kiểu dữ liệu, toán tử đến các cấu trúc điều khiển, cú pháp hay một số thư viện. Lập trình viên Java hay C có thể sử dụng ngay JavaScript với một số lưu ý về những điểm khác biệt giữa các ngôn ngữ này. Do vậy, với giả thiết người học đã biết về Java hay C, giáo trình chỉ trình bày những điểm khác biệt của JavaScript. Những đặc điểm ngôn ngữ của JavaScript không được trình bày trong giáo trình được hiểu là giống với Java.

4.1.1. Định kiểu không tường minh

JavaScript sử dụng cơ chế **định kiểu không tường minh** (implicit typing), hay còn gọi là **định kiểu động** (dynamic typing). Định kiểu không tường minh không phải hiếm trong các ngôn ngữ lập trình. Ngoài JavaScript, các ngôn ngữ lập trình khác như Lisp, Smalltalk, Perl, Python, Ruby, PHP cũng sử dụng định kiểu không tường minh. Theo cơ chế định kiểu này, kiểu được xác định trong thời gian thực thi, kiểu của một biến được xác định bằng kiểu của giá trị đầu tiên gán cho biến, kiểu của hàm được xác định bằng kiểu của giá trị trả về của hàm, và kiểu của tham số được xác định bằng kiểu của giá trị truyền cho chúng.

4.1.2. Biến, hàm

Trong JavaScript, hàm được khai báo bằng từ khóa *function*, biến được khai báo với từ khóa *var* hoặc không cần dùng từ khóa. Ví dụ khai báo hàm và biến như sau.

```
1. <script type="text/javascript">
2.   function cong(x, y) {
3.     return (x+y);
4.   }
5.   var a = 10;
6.   b = 10.5;
7.   c = cong(a, b);
8.   document.write(c);  // 20.5
9. </script>
```

Trong ví dụ này, các biến *a, b, c* cùng hàm *cong()* đã được khai báo. Theo cơ chế định kiểu không tường minh, *a* có kiểu số nguyên (integer), *b, c* và *cong()* có kiểu số thực (float).

4.1.3. Mảng

Mảng trong JavaScript được hiểu là sưu tập (collection) các biến cùng tên và khác chỉ số[12]. Các biến trong cùng mảng có thể khác nhau về kiểu. Mỗi biến có thể nhận kiểu dữ liệu bất kỳ từ các kiểu nguyên thủy như số, xâu, ký tự đến các kiểu phức hợp như đối tượng, thậm chí mảng. Về bản chất, mảng trong JavaScript là ánh xạ với giá trị khóa (chỉ số) là số nguyên. Truy cập các phần tử trong mảng được thực hiện thông qua tên mảng và chỉ số của phần tử. Trong mã nguồn ví dụ sau đây, một mảng được sử dụng để lưu một bản ghi của thí sinh với các thông tin bao gồm họ tên, giới tính, ngày sinh, và ba điểm thành phần. Dễ thấy các phần tử trong mảng có các kiểu dữ liệu khác nhau và truy cập các phần tử mảng, thêm mới cùng thay đổi giá trị các phần tử mảng được thực hiện thông qua tên mảng cùng chỉ số của phần tử.

```
1. <script type="text/javascript">
2.   var a = ["Hoàng Ngân", 'M', new Date('1998-3-18'), [5, 9, 7]];
3.   a[100] = "CLC";              // Thêm mới
4.   document.write(a.length);    // 101
5.   document.write(a[1]);        // M
6.   a[1] = 'F';                  // Thay đổi giá trị
7.   document.write(a[1]);        // F
8.   document.write(a[2]);        // Wed Mar 18 1998 07:00:00 GMT+0700 (ICT)
9.   document.write(a[3][1]);     // 9
10.  document.write(a[15]);       // undefined
11. </script>
```

Cũng có thể sử dụng đối tượng *Array* để khai báo mảng trong JavaScript. Tuy nhiên, lập trình viên được khuyến cáo sử dụng khai báo mảng kiểu nguyên thủy như được minh họa trong ví dụ trên.

[12] Các biến cùng mảng trong C còn có thêm hai ràng buộc khác là cùng kiểu dữ liệu và được cấp phát bộ nhớ liên tiếp nhau.

4.1.4. Đối tượng và kế thừa

Mọi thực thể trong JavaScript đều là đối tượng. Đối tượng là sưu tập (collection) hay là kết hợp động (dynamic association) của các thuộc tính và phương thức. Đối tượng cũng bao gói (encapsulate) dữ liệu (thể hiện bằng thuộc tính) và hành động (thể hiện bằng phương thức). Tuy nhiên, ở mọi thời điểm có thể thêm hoặc bớt các thuộc tính và phương thức bất kỳ cho đối tượng. Trong ví dụ sau đây, đối tượng *person* được khai báo với các thuộc tính *fullname* và *alias* cùng phương thức *sayHello()*. Lưu ý từ khóa *this* được sử dụng trong phương thức *sayHello()* để truy cập các thuộc tính. Cũng lưu ý về sự bình đẳng trong cách khai báo thuộc tính và phương thức. Thuộc tính (dữ liệu) và phương thức (hàm) đều là những thành viên (member) của đối tượng nên có cùng một cách khai báo là *tên_thành_viên:giá_trị*. Sau khi được khai báo, đối tượng *person* được thêm thuộc tính *dob* và bỏ thuộc tính *alias*.

```
1. <script type="text/javascript">
2.   var person = {
3.     fullname: "Hoàng Tùng",
4.     alias: "Bolero",
5.     sayHello: function() {
6.       document.write(this.fullname + " " + this.alias);
7.     }
8.   };
9.
10.  person.sayHello();              // Hoàng Tùng Bolero
11.  person.dob = "22/2/2012";
12.  delete person.alias;
13.  person.sayHello();              // Hoàng Tùng undefined
14.  document.write(person.dob);     // 22/2/2012
15. </script>
```

Có thể sử dụng lớp *Object* được dựng sẵn trong JavaScript để khai báo đối tượng, sau đó lần lượt thêm các thuộc tính và phương thức cho đối tượng như ví dụ sau. Cách khai báo này và cách khai báo ở ví dụ trước là tương đương nhau.

```
1. <script type="text/javascript">
2.   var person = new Object();
3.   person.fullname = "Hoàng Tùng";
4.   person.alias = "Bolero";
5.   person.sayHello = function() {
6.     document.write(this.fullname + " " + this.alias);
7.   }
8.
9.   person.sayHello();              // Hoàng Tùng Bolero
10.  person.dob = "22/2/2012";
11.  delete person.alias;
12.  person.sayHello();              // Hoàng Tùng undefined
13.  document.write(person.dob);     // 22/2/2012
14. </script>
```

Các phương pháp khai báo đối tượng ở trên chỉ phù hợp khi cần khai báo một vài đối tượng đơn lẻ. Nếu cần khai báo hàng loạt các đối tượng có kiểu giống nhau, JavaScript cho định nghĩa hàm tạo (constructor) và sử dụng hàm tạo để khai

báo đối tượng. Lưu ý, JavaScript không có định nghĩa lớp (class) giống như Java hay C++. Ví dụ khai báo và sử dụng hàm tạo như sau.

```
1. <script type="text/javascript">
2.   function Person(fn, al) {
3.       this.fullname = fn;
4.       this.alias = al;
5.       this.sayHello = function() {
6.           document.write(this.fullname + " " + this.alias);
7.       }
8.   }
9.   person = new Person("Hoàng Tùng", "Bolero");
10.  person.sayHello(); // Hoàng Tùng Bolero
11.  papa = new Person("Hoàng Bách", "Sava");
12.  papa.sayHello();   // Hoàng Bách Sava
13. </script>
```

Sử dụng hàm tạo không chỉ có ưu điểm là cho khai báo hàng loạt các đối tượng cùng kiểu như ví dụ trên mà còn cho phép sửa đổi tất cả các đối tượng bằng một sửa đổi ở hàm tạo. Trong ví dụ sau đây, hàm tạo *Person* được bổ sung phương thức *sayGoodbye()*, do vậy tất cả các đối tượng đều có thêm thành viên hàm *sayGoodbye()*.


```
1. <script type="text/javascript">
2.   function Person(fn, al) {
3.     this.fullname = fn; this.alias = al;
4.     this.sayHello = function() {
5.       document.write(this.fullname + " " + this.alias);
6.     }
7.   }
8.   person = new Person("Hoàng Tùng", "Bolero");
9.   person.sayHello();        // Hoàng Tùng Bolero
10.  papa = new Person("Hoàng Bách", "Sava");
11.  papa.sayHello();          // Hoàng Bách Sava
12.
13.  Person.prototype.sayGoodbye = function () {
14.    document.write(this.fullname +  " good bye everyone!");
15.  };
16.  person.sayGoodbye();      // Hoàng Tùng good bye everyone!
17.  papa.sayGoodbye();        // Hoàng Bách good bye everyone!
18. </script>
```

Trong thực hành, để trình bày mã nguồn được sáng sủa, hàm tạo được khuyến cáo chỉ thiết lập các thuộc tính, còn các phương thức được thêm sau như cách *sayGoodbye()* được thêm vào *Person* ở ví dụ trên.

Thuộc tính và phương thức được khai báo với từ khóa *this* có phạm vi truy cập là *public*. Ngược lại, thuộc tính và phương thức được khai báo trong hàm tạo mà không có từ khóa *this* sẽ có phạm vi truy cập là *private*. Trong ví dụ sau đây, hàm tạo *Person* được định nghĩa lại với các thuộc tính *fullname* và *alias* có phạm vi truy cập là *private*, đồng thời phương thức *getAllNames()* được bổ sung cũng có phạm vi truy cập là *private*. Lưu ý, việc truy cập các thuộc tính *private* không có từ khóa *this* ở trước. Cũng lưu ý việc truy cập phương thức *private* từ phương thức *public* được

thông qua phương thức *apply()* với con trỏ *this*. Câu lệnh cuối trong chương trình có lỗi do vi phạm phạm vi truy cập của *getAllNames()*.

```
1. <script type="text/javascript">
2.   function Person(fn, al) {
3.     var fullname = fn;
4.     var alias = al;
5.     function getAllNames() {return (fullname + " " + alias);}
6.     this.sayHello = function() {
7.       document.write(getAllNames.apply(this));
8.     }
9.   }
10.  person = new Person("Hoàng Tùng", "Bolero");
11.  person.sayHello();    // Hoàng Tùng Bolero
12.  person.getAllNames(); // Lỗi
13. </script>
```

Kế thừa trong JavaScript được thực hiện dựa trên nguyên mẫu (prototype-based). Mỗi đối tượng trong JavaScript tham chiếu đến một đối tượng khác, gọi là đối tượng nguyên mẫu, và kế thừa các thuộc tính, phương thức của đối tượng nguyên mẫu. Nguyên mẫu của các đối tượng *Object* là *null*. Nếu không có chỉ định rõ ràng, nguyên mẫu của đối tượng mặc định là *Object.prototype*. Các phương thức *Object.create()* và *Object.getPrototypeOf()* được sử dụng để tạo và lấy nguyên mẫu. Đối tượng nguyên mẫu lại có nguyên mẫu của nó. Quan hệ nguyên mẫu tạo nên **chuỗi nguyên mẫu** (prototype chain). Trong ví dụ sau đây, đối tượng *faculty* có nguyên mẫu là đối tượng *person*. Ngoài những thuộc tính và phương thức được kế thừa từ *person*, *faculty* còn có các thuộc tính và phương thức riêng của nó.


```
1. <script type="text/javascript">
2.   function Person(fn, al) {
3.     var fullname = fn;
4.     var alias = al;
5.     function getAllNames() {
6.       return (fullname + " " + alias);
7.     }
8.     this.sayHello = function() {
9.       document.write(getAllNames.apply(this));
10.    }
11.  }
12.
13.  person = new Person("Hoàng Tùng", "Bolero");
14.  faculty  = Object.create(person);
15.  faculty.department = "Khoa CNTT";
16.
17.  faculty.sayHello = function() {
18.    person.sayHello();
19.    document.write(" " + this.department);
20.  };
21.
22.  faculty.sayHello(); // Hoàng Tùng Bolero Khoa CNTT
23. </script>
```

Sử dụng hàm tạo là phương thức chính được sử dụng để tạo chuỗi nguyên mẫu/kế thừa cho các đối tượng. Trong ví dụ sau đây, hàm tạo *Staff* được kế thừa

nguyên mẫu từ hàm tạo *Person* bằng lời gọi *Person.call()*. Mỗi đối tượng được khai báo bằng *Staff* sẽ có nguyên mẫu là một đối tượng *Person*. Tất cả các thuộc tính và phương thức từ *Person* được kế thừa sang *Staff*. Lưu ý, *Staff* không thể truy cập các thành viên *private* của *Person*. Trong ví dụ đang xét, phương thức *sayGoodbye()* của *Staff* cần truy cập thuộc tính *fullname* nhưng phải thực hiện thông qua phương thức *getFullname()* được kế thừa và có phạm vi truy cập *public*. Cũng lưu ý, *Staff* có thể định nghĩa chồng (overriding) phương thức của *Person* như phương thức *sayHello()* trong ví dụ.

```
1. <script type="text/javascript">
2.   function Person(fn, al) {
3.     var fullname = fn;
4.     var alias = al;
5.     function getAllNames() {return (fullname + " " + alias);}
6.     this.sayHello = function() {
7.       document.write(getAllNames.apply(this));
8.     };
9.     this.getFullname = function() {return fullname;};
10.  }
11.
12.  function Staff(fn, al, sa) {
13.    Person.call(this, fn, al);
14.    var salary = sa;
15.    var parentHello = this.sayHello;
16.    this.sayHello = function() {
17.      parentHello.apply(this);
18.      document.write(" with salary " + salary);
19.    };
20.    this.sayGoodbye = function() {
21.         document.write(this.getFullname() + " good bye everyone!");
22.    };
23.  }
24.
25.  var staff = new Staff("Hoàng Ngân", "Diamon", 1000);
26.  staff.sayHello();   // Hoàng Ngân Diamon with salary 1000
27.  staff.sayGoodbye(); // Hoàng Ngân good bye everyone!
28. </script>
```

Phiên bản JavaScript ECMAScript 2015 giới thiệu các từ khóa *class*, *constructor*, *extends* và *super* để định nghĩa nguyên mẫu và kế thừa. Tuy nhiên, các trình duyệt chưa hỗ trợ đầy đủ các từ khóa này. Do vậy, cho đến nay, sử dụng hàm tạo để khai báo và sử dụng đối tượng trong JavaScript vẫn là phương thức chính tắc được sử dụng rộng rãi.

4.1.5. Xâu ký tự

Giá trị nguyên thủy (literal) của xâu ký tự trong JavaScript là chuỗi ký tự nằm giữa hai dấu nháy đơn (') hoặc giữa hai dấu nháy kép ("). Bản thân mỗi xâu ký tự trong JavaScript, kể cả xâu với giá trị nguyên thủy, là một đối tượng. Đối tượng xâu trong JavaScript cung cấp nhiều phương thức xử lý xuất phát từ yêu cầu quản lý các đối tượng tài liệu thuộc trang web và tất cả thuộc tính của chúng đều có giá

trị kiểu xâu. Một số phương thức và thuộc tính của xâu thường xuyên được sử dụng được liệt kê và mô tả dưới đây:

s.length	Độ dài của xâu *s*.
s[i]	Ký tự có chỉ mục *i* trong xâu *s*.
s.indexOf(s1)	Tìm vị trí xuất hiện đầu tiên của xâu con *s1* trong xâu *s*.
s.replace(s1, s2)	Thay thế các xâu con *s1* trong xâu *s* bằng xâu *s2*.
s.substring(b, e)	Xâu con của *s* bao gồm các ký tự có chỉ mục từ *b* đến *e*-1.
s.substring(b)	Xâu con của *s* bao gồm các ký tự có chỉ mục từ *b* đến hết.
s.split(d)	Tách xâu *s* bởi xâu ngăn cách *d*.
s.toUpperCase()	Trả về xâu viết hoa của *s*.
s.toLowerCase()	Trả về xâu viết thường của *s*.

Ngoài ra, JavaScript cung cấp một số hàm kiểm tra và chuyển đổi kiểu từ xâu thành số như liệt kê dưới đây:

isNaN(s)	*true* nếu *s* không là biểu diễn số.
parseInt(s)	Giá trị nguyên của biểu diễn *s*.
parseFloat(s)	Giá trị thực của biểu diễn *s*.

4.2. MÔ HÌNH ĐỐI TƯỢNG TÀI LIỆU

DOM (Document Object Model)[13] là chuẩn của W3C được sử dụng để truy cập các tài liệu, trong đó HTML DOM là chuẩn mô hình đối tượng và giao diện lập trình cho HTML. DOM định nghĩa các đối tượng, thuộc tính, phương thức và sự kiện (event) trên các đối tượng. Nói cách khác, DOM là chuẩn cho việc đọc, thay đổi, thêm mới hay loại bỏ các đối tượng tài liệu thuộc trang web.

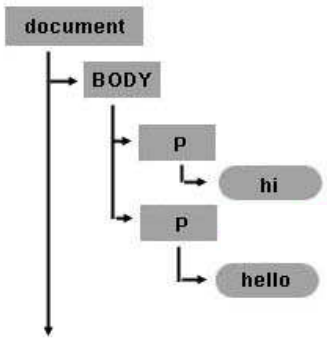
Khi trình duyệt thông dịch và chạy mã nguồn trang web, các đối tượng tài liệu được tạo và lưu trữ trong bộ nhớ theo cấu trúc cây, được gọi là cây DOM (DOM tree). Gốc của cây DOM là đối tượng *document*. Đối tượng *document* đại diện cho trang web. Tất cả các đối tượng được khai báo bằng HTML đều thuộc dòng dõi (con, cháu, …) của *document*. Các đối tượng được khai báo bằng các thẻ lồng nhau có quan hệ cha-con (parent-child) với nhau, trong đó đối tượng được khai báo bằng thẻ nằm bên trong là đối tượng con. Các đối tượng được khai báo liên tiếp nhau có quan hệ anh-em (sibling) với nhau. Dễ dàng nhận thấy ánh xạ một-một giữa khai báo HTML và cây DOM. Từ mã nguồn HTML có thể suy ra cây DOM,

[13] Bao gồm Core DOM, HTML DOM và XML DOM. Trong giáo trình này, nếu không có chú thích rõ ràng, DOM được hiểu là HTML DOM.

và ngược lại từ cây DOM có thể viết lại HTML đã khai báo ra nó. Tương quan giữa khai báo HTML và cây DOM được minh họa trong một vài ví dụ sau đây. Ví dụ thứ nhất, giả sử trang web có mã nguồn là:

```
1. <html>
2.  <body>
3.     <p>hi</p>
4.     <p>hello</p>
5.   </body>
6. </html>
```

Cây DOM của trang web sẽ có dạng như Hình 4.1.

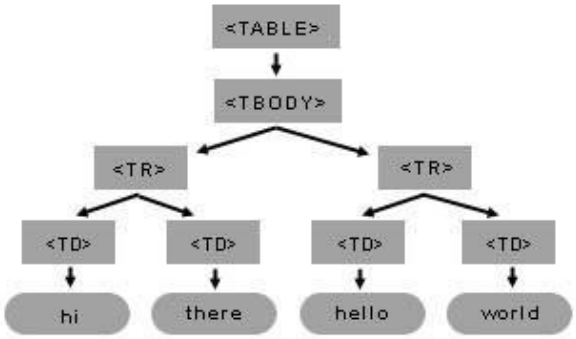


Hình 4.1. Cây DOM của một trang web.

Một ví dụ khác, khai báo đối tượng bảng với mã nguồn như sau:

```
1. <table>
2.   <tbody>
3.     <tr>
4.       <td> hi </td>
5.       <td> there </td>
6.     </tr>
7.     <tr>
8.       <td> hello </td>
9.       <td> world </td>
10.    </tr>
11.  </tbody>
12.</table>
```

DOM được tạo cho đối tượng bảng ở ví dụ trên sẽ có dạng như Hình 4.2.



Hình 4.2. DOM cho một đối tượng bảng.

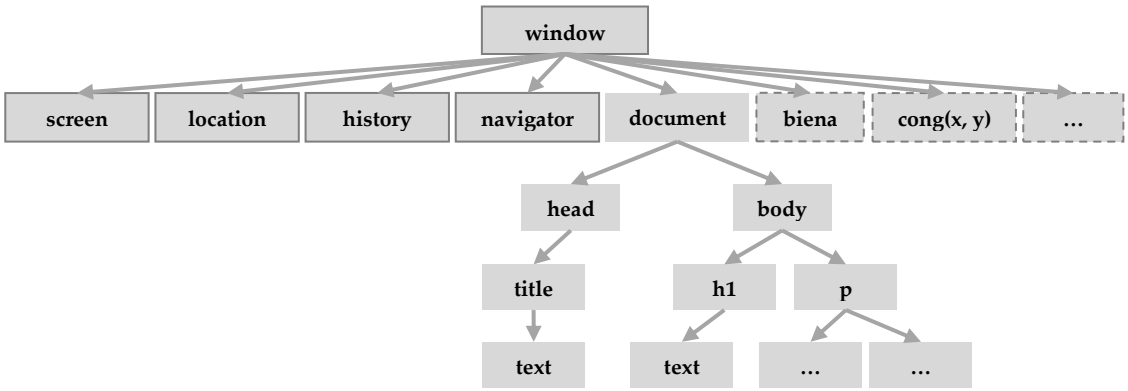
Mỗi đối tượng, còn gọi là một nút (node), có các thuộc tính sau đây thể hiện quan hệ giữa nó với những đối tượng (nút) khác thuộc cùng cây DOM:

parentNode	Tham chiếu đến nút cha.
childNodes	Mảng các tham chiếu đến các nút con.
firstChild	Tham chiếu đến nút con đầu tiên.
lastChild	Tham chiếu đến nút con cuối cùng.
previousSibling	Tham chiếu đến nút liền trước.
nextSibling	Tham chiếu đến nút liền sau.

Từ góc độ kỹ thuật, trang web là cây DOM và nhiệm vụ của ứng dụng web là tạo ra, duy trì và cập nhật cây DOM. Những thay đổi trên cây DOM sẽ được phản ánh ra trình diễn của trang web trên giao diện của trình duyệt, tương tự thay đổi nội dung dòng ra chuẩn (standard output) sẽ làm thay đổi hiển thị trên màn hình của máy tính.

4.3. MÔ HÌNH ĐỐI TƯỢNG TRÌNH DUYỆT

BOM (Browser Object Model) cho phép JavaScript tương tác với trình duyệt. Mặc dù không có chuẩn cho BOM nhưng hầu hết các trình duyệt hiện đại đều cài đặt các phương thức và thuộc tính BOM như nhau. Các đối tượng BOM cũng có quan hệ với nhau theo cấu trúc cây. Gốc của cây BOM là đối tượng *window*. *window* đại diện cho cửa sổ trình duyệt. Tất cả các đối tượng và biến toàn cục JavaScript đều trở thành thuộc tính của *window*. Tương tự, tất cả các hàm JavaScript được khai báo toàn cục sẽ tự động trở thành phương thức của *window*. Thậm chí, đối tượng *document* của DOM cũng trở thành một thuộc tính của *window*. Nói cách khác, cây DOM sẽ trở thành một nhánh của cây BOM. Truy cập các thành viên (thuộc tính và phương thức) của đối tượng *window* có thể theo chính tắc hoặc không chính tắc chỉ cần viết tên thành viên. Ví dụ, *window.document* tương đương *document*, *window.cong(10, 3)* tương đương *cong(10, 3)*, và *window.biena* tương đương *biena*, với *cong(x, y)* là một hàm và *biena* là một biến JavaScript được khai báo toàn cục. Quan hệ giữa *window* và các đối tượng còn lại, cũng như quan hệ giữa BOM với DOM được minh họa trong Hình 4.3. Trong hình minh họa này, mỗi đối tượng DOM được biểu diễn bằng một hình chữ nhật màu xám không có đường viền, mỗi đối tượng BOM dựng sẵn được biểu diễn bằng một hình chữ nhật màu xám có viền nét liền, và mỗi biến hay phương thức JavaScript được khai báo toàn cục được biểu diễn bằng một hình chữ nhật màu xám có viền nét đứt.



Hình 4.3. BOM và DOM.

Đối tượng *window* có các phương thức *resizeTo(x, y), resizeBy(dx, dy)* để thay đổi kích thước cửa sổ, các phương thức *scrollTo(x, y), scrollBy(dx, dy)* để thay đổi vị trí thanh cuộn. Ngoài ra, *window* có các phương thức *alert(msg), confirm(msg), open(url)* để hiển thị thông báo hay mở cửa sổ mới, *print()* để in nội dung đang được hiển thị trên cửa sổ. Một số đối tượng con dựng sẵn quan trọng của *window* được mô tả dưới đây.

Đối tượng *window.**screen*** mang thông tin về màn hình của máy người dùng. Đối tượng này có các thuộc tính *width, height* cho biết độ rộng, cao của màn hình tính theo pixel, các thuộc tính *availWidth, availHeight* cho biết độ rộng, cao khả dụng của màn hình tính theo pixel, thuộc tính *colorDepth* cho biết số bit được sử dụng để biểu diễn một giá trị màu.

Đối tượng *window.**location*** cho biết URL của trang hiện tại và có thể được sử dụng để chuyển hướng cửa sổ đến trang mới. Nó có các thuộc tính *href, hostname, pathname, protocol* cho biết URL, tên miền, đường dẫn, giao thức trong URL của trang hiện tại, tương ứng. Phương thức *assign(url)* được sử dụng để tải tài nguyên mới vào cửa sổ.

Đối tượng *window.**history*** lưu lịch sử duyệt của cửa sổ. Để bảo vệ tính riêng tư của người dùng, đối tượng này hạn chế cho sử dụng JavaScript để truy cập đến nó. Theo đó, nó chỉ cung cấp các phương thức *back()* và *forward()* để chuyển sang tài nguyên trước hoặc sau tài nguyên hiện tại trong lịch sử duyệt.

Đối tượng *window.**navigator*** mang thông tin về trình duyệt của người dùng. Thuộc tính *cookieEnabled* cho biết trình duyệt có sử dụng cookies hay không. Các thuộc tính *appName*, *appCodeName*, *appVersion* cho biết tên, mã và phiên bản của trình duyệt. Thuộc tính *platform* cho biết hệ điều hành chạy trình duyệt. Lưu ý, không nên sử dụng thông tin trong *navigator* để phát hiện phiên bản trình duyệt vì thông tin có thể không chính xác hoặc bị giả mạo.

Đối tượng *window.**document*** đặc biệt quan trọng vì nó là gốc của cây DOM. Nó đại diện cho trang web được tải vào cửa sổ. Sử dụng *document* để truy cập tất cả

đối tượng tài liệu được khai báo trong trang web. Sử dụng đối tượng *document* sẽ được trình bày trong các mục tiếp theo.

4.4. QUẢN LÝ TRANG WEB

4.4.1. Lấy tham chiếu các đối tượng tài liệu

Bước đầu tiên trong mọi thao tác quản lý trang web là lấy tham chiếu (reference) đến các đối tượng tài liệu. Khi đã lấy được tham chiếu, chương trình có thể sử dụng tham chiếu để truy cập và thay đổi các thuộc tính cũng như triệu gọi các phương thức của đối tượng. Về cơ bản, có thể duyệt cây DOM để lấy tham chiếu của đối tượng bất kỳ. Tuy nhiên, thao tác duyệt như vậy sẽ hết sức tốn kém. Thay vào đó, đối tượng *document*, gốc của cây DOM, cung cấp nhiều phương thức lấy tham chiếu đối tượng một cách dễ dàng và hiệu quả, như được trình bày sau đây.

Phương thức *document.**getElementById**(v)* trả về tham chiếu đối tượng tài liệu có định danh (thuộc tính *id*) là *v*. Trường hợp trang web không chứa đối tượng có định danh *v*, phương thức *document.getElementById(v)* sẽ trả về *null*.

Phương thức *document.**getElementsByName**(v)* trả về mảng tham chiếu các đối tượng tài liệu có tên (thuộc tính *name*) là *v*. Sử dụng thủ tục duyệt mảng để truy cập từng đối tượng được tham chiếu.

Phương thức *document.**getElementsByTagName**(v)* trả về mảng tham chiếu các đối tượng tài liệu được khai báo bằng thẻ *v*. Sử dụng thủ tục duyệt mảng để truy cập từng đối tượng được tham chiếu.

Phương thức *document.**querySelector**(v)* trả về tham chiếu đối tượng đầu tiên được chọn bởi bộ chọn CSS *v*. Trường hợp trang web không chứa đối tượng được chọn bởi *v*, phương thức *document.querySelector(v)* sẽ trả về *null*.

Phương thức *document.**querySelectorAll**(v)* trả về mảng tham chiếu các đối tượng được chọn bởi bộ chọn CSS *v*. Sử dụng thủ tục duyệt mảng để truy cập từng đối tượng được tham chiếu.

Khi đã có tham chiếu đến một đối tượng, giả sử *obj*, có thể dễ dàng lấy tham chiếu đến đối tượng cha cũng như các đối tượng con, anh và em của nó thông qua các thuộc tính *obj.parentNode*, *obj.childNodes*, *obj.previousSibling* và *obj.nextSibling* tương ứng.

Ví dụ lấy tham chiếu đối tượng như sau. Trong ví dụ này, đối tượng danh sách không thứ tự được lấy tham chiếu theo định danh, các mục có thứ tự lẻ trong danh sách được lấy tham chiếu theo bộ chọn CSS[14].

[14] Hàm *console.log(msg)* xuất nội dung *msg* ra dòng ra chuẩn. Từ các trình duyệt có thể xem nội dung này trong cửa sổ Web Console thuộc nhóm công cụ cho nhà phát triển.

```
1. <ul id="mylist">
2.   <li>Item 1</li>
3.   <li>Item 2</li>
4.   <li>Item 3</li>
5. </ul>
6.
7. <script type="text/javascript">
8.   var list = document.getElementById("mylist");
9.   console.log(list.innerHTML); // <li>Item 1</li><li>Item 2</li><li>Item 3</li>
10.  var oddi = document.querySelectorAll("#mylist li:nth-of-type(odd)");
11.  for (var i = 0; i < oddi.length; i++)
12.    console.log(oddi[i].innerHTML); // Item 1   Item 3
13. </script>
```

4.4.2. Đọc và thay đổi giá trị thuộc tính của đối tượng tài liệu

Khi đã lấy được tham chiếu của một đối tượng tài liệu, việc đọc hay thay đổi thuộc tính của đối tượng trở nên bình thường. Trong ví dụ sau đây, giá trị (thuộc tính *value*) của ô nhập chữ có định danh *fullname* được kiểm tra. Nếu giá trị là trống, tức người dùng chưa nhập, lời nhắc được hiển thị bằng cách đặt nội dung lời nhắc cho thuộc tính *innerHTML* của đối tượng ** có định danh là *fullnameErr*.


```
1. <label for="fullname">Họ và tên:</label>
2. <input type="text" id="fullname"/>
3. <span id="fullnameErr"></span>
4. <script type="text/javascript">
5.     var fn = document.getElementById("fullname");
6.     var fnerr = document.getElementById("fullnameErr");
7.     fnerr.innerHTML = (fn.value == "" ? "Chưa nhập họ tên" : "");
8. </script>
```

Có thể liệt kê tất cả các thuộc tính của đối tượng tài liệu bằng thuộc tính *attributes*. Thuộc tính này trả về mảng chứa tất cả thuộc tính của đối tượng. Ví dụ sau duyệt và in ra tất cả các thuộc tính của đối tượng nhập chữ.

```
1. <input type="text" id="fullname" value="Trần Văn Nguyên"/>
2.
3. <script type="text/javascript">
4.   var attrs = document.getElementById("fullname").attributes;
5.   for (var i = 0; i < attrs.length; i++) {
6.     console.log(attrs[i].name + ": " + attrs[i].value);
7.     //id: fullname   value: Trần Văn Nguyên   type: text
8.   }
9. </script>
```

4.4.3. Thay đổi kiểu trình diễn đối tượng tài liệu

Đối tượng tài liệu có các thuộc tính *style*, *className* và *classList* để lưu trữ kiểu trình diễn cho nó. Đọc và thay đổi giá trị của các thuộc tính này cũng giống các thuộc tính khác. Bản thân *style* là một bộ sưu tập với nhiều thuộc tính trình diễn bên trong nó. Sử dụng *style* có thể truy cập và thay đổi từng thuộc tính trình diễn cụ thể. Mặt khác, thuộc tính *className* lưu danh sách tên các bộ chọn CSS đã chọn

đối tượng tài liệu. Sử dụng *className* có thể thay đổi các bộ chọn CSS cho đối tượng tài liệu. Ngoài ra, *classList* cung cấp các giao diện lập trình *add(v1, v2, …)*, *remove(v1, v2, …)*, *toggle(v, true|false)*, *contains(v)* để thêm, bớt, thay đổi hiệu lực và kiểm tra việc sử dụng các lớp CSS một cách đơn giản hơn. Trang web ví dụ sau đây là nâng cấp của trang web ở ví dụ trước. Ngoài việc hiển thị lời nhắc, trang web mới còn thay đổi kiểu trình diễn lời nhắc.

```
1.  <!DOCTYPE html><html><head><meta charset="utf-8">
2.    <style type="text/css">
3.       .err {color:red;} .required {font-weight:bold;}
4.       .critical {font-size:x-large;}
5.       .highlight {border:solid 1px green;}
6.    </style>
7.  </head><body>
8.    <label for="fullname">Họ và tên:</label>
9.    <input type="text" id="fullname"/>
10.   <span id="fullnameErr"></span>
11.   <script type="text/javascript">
12.      var fn = document.getElementById("fullname");
13.      var fnerr = document.getElementById("fullnameErr");
14.      if (fn.value == "") {
15.        fnerr.innerHTML = "Chưa nhập họ tên";
16.        fnerr.style.backgroundColor = "yellow";
17.        fnerr.className = "err required";
18.        fnerr.classList.add("critical", "highlight");
19.      }
20.   </script>
21. </body></html>
```

4.4.4. Xử lý sự kiện trên đối tượng tài liệu

Tất cả các đối tượng tài liệu khi được hiển thị trên giao diện của trình duyệt đều có thể bắt các sự kiện (chuột và bàn phím) khi sự kiện xảy ra trên nó. Ứng với mỗi sự kiện, đối tượng có một thuộc tính là con trỏ sự kiện/hàm lưu tham chiếu tới hàm JavaScript sẽ được gọi khi sự kiện xảy ra. Một số con trỏ sự kiện quan trọng, chung cho tất cả các đối tượng tài liệu, được liệt kê và mô tả dưới đây.

Sự kiện	**Ý nghĩa**
onclick	Được gọi khi người dùng kích chuột vào đối tượng.
ondblclick	Được gọi khi người dùng kích đúp chuột vào đối tượng.
onmouseover	Được gọi khi người dùng đưa chuột lên đối tượng.
onmouseout	Được gọi khi người dùng đưa chuột ra khỏi đối tượng.
onfocus	Được gọi khi đối tượng được đặt tâm điểm.
onblur	Được gọi khi đối tượng mất tâm điểm.
onkeydown	Được gọi khi đối tượng đang được đặt tâm điểm và người dùng nhấn phím trên bàn phím.

onkeyup Được gọi sau *onkeydown* khi người dùng nhả phím.

Trang web ví dụ sau đây minh họa hai phương thức cơ bản để gán hàm xử lý sự kiện cho đối tượng. Phương thức thứ nhất là gán giá trị (hàm JavaScript) cho con trỏ sự kiện cùng với khai báo HTML. Ví dụ, đối tượng nút bấm thứ nhất, có nhãn là "Chấp nhận", được gán hàm xử lý sự kiện *onclick="checkInput();"* cùng với khai báo. Phương thức thứ hai là gán giá trị cho con trỏ sự kiện sau khai báo HTML. Ví dụ, đối tượng nút bấm thứ hai, có nhãn là "Bỏ qua", được gán hàm xử lý sự kiện *onclick* bằng một hàm JavaScript nội tuyến, không tên. Với trang web này, mỗi khi người dùng kích chuột vào nút "Chấp nhận", hàm *checkInput()* sẽ được gọi để kiểm tra giá trị nhập và hiển thị thông báo lỗi nếu có; ngược lại nếu người dùng kích chuột vào nút "Bỏ qua", cửa sổ sẽ chuyển sang trang *Page2.htm*.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.4.4.4-1</title>
3.   <meta charset="utf-8">
4.   <style type="text/css"> .err {color:red;} </style>
5. </head><body>
6.   <label for="fullname">Họ và tên:</label>
7.   <input type="text" id="fullname"/>
8.   <span id="fullnameErr" class="err"></span>
9.   <br/>
10.  <button onclick="checkInput();">Chấp nhận</button>
11.  <button id="btncancel">Bỏ qua</button>
12.  <script type="text/javascript">
13.    function checkInput() {
14.      var fn = document.getElementById("fullname");
15.      var fnerr = document.getElementById("fullnameErr");
16.      if (fn.value == "") {
17.        fnerr.innerHTML = "Chưa nhập họ tên";
18.      } else {
19.        // đệ trình ở đây
20.      }
21.    }
22.    document.getElementById("btncancel").onclick = function() {
23.      window.location = "Page2.htm";
24.    }
25.  </script>
26. </body></html>
```

Ngoài các con trỏ sự kiện chung như được liệt kê ở trên, một vài đối tượng tài liệu có thể có các con trỏ sự kiện riêng. Ví dụ, đối tượng *<body>* có các sự kiện *onload* (xảy ra lúc tải trang) và *onunload* (xảy ra lúc rời trang), đối tượng *<select>* có sự kiện *onchange* (xảy ra khi người dùng thay đổi mục chọn), đối tượng *<form>* có sự kiện *onsubmit*, …

Cũng như bất kỳ phương thức nào khác, phương thức được gán cho con trỏ sự kiện có thể sử dụng từ khóa *this* để tham chiếu đến đối tượng sở hữu nó. Trong trang web ví dụ sau đây, hàm xử lý sự kiện nhả phím trên ô nhập "Họ và tên" sử dụng *this* để tham chiếu đến đối tượng nhập chữ có định danh là *fullname*. Hàm này gọi đến hàm *checkInput()*, đồng thời truyền *this* cho nó, để kiểm tra dữ liệu

nhập trống hay khác trống. Nếu dữ liệu nhập khác trống, nền của ô nhập được đặt màu vàng, ngược lại nền của ô nhập được đặt màu trắng.

```
1. <!DOCTYPE html><html><head>
2.     <title>L.4.4.4-2</title>
3.     <meta charset="utf-8">
4. </head><body>
5.     <label for="fullname">Họ và tên:</label>
6.     <input type="text" id="fullname"/>
7.     <br/>
8.     <label>Địa chỉ:</label>
9.     <input type="text" id="address"/>
10.    <script type="text/javascript">
11.        function checkInput(ctrl) {
12.            return (ctrl.value == "" ?  false : true);
13.        }
14.        document.getElementById("fullname").onkeyup = function(e) {
15.            if (!checkInput(this)) this.style.backgroundColor = "white";
16.            else this.style.backgroundColor = "yellow";
17.
18.            var kc = window.event ? window.event.keyCode : e.keyCode;
19.            if (kc == 13) { // Enter
20.                document.getElementById("address").focus();
21.            }
22.        };
23.    </script>
24. </body></html>
```

Một lưu ý nữa khi gán hàm xử lý sự kiện cho đối tượng tài liệu là có thể truyền cho hàm một tham số là chính đối tượng sự kiện. Đối tượng sự kiện mang thông tin về sự kiện. Trong trang web ví dụ ở trên, hàm xử lý sự kiện nhả phím trên ô nhập "Họ và tên" nhận đối tượng sự kiện bằng tham số *e*. Đối tượng *e* trong ví dụ này lưu giữ các thông tin về sự kiện nhả phím. Hàm sự kiện kiểm tra phím nhả có phải là Enter hay không, nếu đúng như vậy thì đặt tâm điểm vào ô nhập tiếp theo, tức ô nhập "Địa chỉ" có định danh là *address*.

Bên cạnh mỗi con trỏ sự kiện, đối tượng tài liệu có một phương thức sự kiện tương ứng. Nếu như con trỏ sự kiện được sử dụng để bắt và xử lý sự kiện như đã được trình bày ở trên, thì phương thức sự kiện được sử dụng để kích hoạt sự kiện. Trong những tình huống cụ thể, thay vì phải chờ người dùng kích hoạt sự kiện, trang web có thể tự động kích hoạt sự kiện bằng cách sử dụng các phương thức sự kiện. Phương thức sự kiện có tên giống tên con trỏ sự kiện nhưng không có tiền tố "on", và không có tham số, ví dụ *click(), focus(), blur(),* … Trong trang web ví dụ sau đây, sự kiện *focus()* trên ô nhập nhập "Họ và tên" được kích hoạt tự động, do vậy ô nhập này được đặt tâm điểm ngay sau khi trang được hiển thị.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.4.4.4-3</title>
3.   <meta charset="utf-8">
4. </head><body>
5.   <label for="fullname">Họ và tên:</label>
6.   <input type="text" id="fullname"/>
7.   <span id="fullnameErr"></span>
```

```
8.    <script type="text/javascript">
9.      document.getElementById("fullname").focus();
10.   </script>
11. </body></html>
```

4.4.5. Thêm mới, loại bỏ đối tượng tài liệu

Cách thức đơn giản nhất để thêm mới hay loại bỏ các đối tượng tài liệu là sử dụng thuộc tính *innerHTML*. Nếu muốn thêm mới đối tượng, hãy thêm nội dung hay khai báo HTML vào *innerHTML* của một đối tượng chứa. Ngược lại, nếu muốn thay đổi hay loại bỏ đối tượng, hãy thay đổi *innerHTML* của đối tượng chứa. Trong trang web ví dụ sau đây, một đoạn văn được thêm vào cuối "Phần 1", đoạn văn thuộc "Phần 2" được thay thế bằng một hình ảnh; tất cả các thay đổi được thực hiện thông qua thuộc tính *innerHTML* của các đối tượng chứa (trong ví dụ là * *).

```
1.  <!DOCTYPE html><html><head>
2.      <title>L.4.4.5-1</title>
3.    <meta charset="utf-8">
4.  </head><body>
5.      <h1>Phần 1</h1>
6.      <div id="container1"><p>Đoạn văn thứ nhất</p></div>
7.      <h1>Phần 2</h1>
8.      <div id="container2"><p>Đoạn văn</p></div>
9.      <script>
10.         var div1 = document.getElementById("container1");
11.         div1.innerHTML += "<p>Đoạn văn thứ hai</p>";
12.         var div2 = document.getElementById("container2");
13.         div2.innerHTML = "<img src='sunset.jpg'>";
14.     </script>
15. </body></html>
```

Sử dụng *innerHTML* có ưu điểm là đơn giản nhưng có hạn chế là chứa đựng rủi ro về an ninh. Nếu nội dung được đưa vào *innerHTML* không được kiểm tra, làm sạch, tấn công vào trang web có thể xảy ra.

Cách thức chính và chuẩn tắc để thêm mới hay loại bỏ các đối tượng tài liệu thuộc trang web là sử dụng các phương thức hay giao diện lập trình của DOM. Đối tượng *document* cung cấp các phương thức tạo đối tượng tài liệu. Tất cả các đối tượng đều cung cấp các phương thức để thêm mới, loại bỏ, thay thế nút con. Chi tiết các phương thức DOM cho việc thêm mới và loại bỏ đối tượng tài liệu thuộc trang web được trình bày tóm tắt dưới đây theo sau là ví dụ về việc sử dụng chúng.

Phương thức	**Mô tả**
document.createElement(tagname)	Tạo đối tượng có tên thẻ *tagname*.
document.createTextNode(txt)	Tạo đối tượng text có nội dung là *txt*.
document.createComment(txt)	Tạo chú thích có nội dung là *txt*.
document.createDocumentFragment()	Tạo phân mảnh tài liệu.

obj.appendChild(node)	Thêm *node* thành con cuối của *obj*.
obj.removeChild(node)	Loại bỏ nút *node* là con của *obj*.
obj.replaceChild(newN, oldN)	Thay nút con *oldN* bằng *newN*.
obj.insertBefore(newN, targetN)	Thêm nút con *newN* trước nút *targetN*.
obj.insertAfter(newN, targetN)	Thêm nút con *newN* sau nút *targetN*.
obj.hasChildNodes()	*true* nếu *obj* có các nút con.

Trong trang web ví dụ sau đây, ba đối tượng đoạn văn, *p3*, *p4*, và *p5* được tạo bằng JavaScript và thêm vào cây DOM, đồng thời thay thế một đối tượng đoạn văn được khai báo trước bằng HTML. Lưu ý, các đối tượng tài liệu được khai báo bằng JavaScript khi chưa được gắn vào cây DOM sẽ không được hiển thị trên giao diện của trình duyệt.

```
1.  <!DOCTYPE html><html><head>
2.    <title>L.4.4.5-2</title>
3.    <meta charset="utf-8"/>
4.  </head><body>
5.    <h1>Phần 1</h1>
6.    <div id="container1">
7.      <p>Đoạn văn thứ nhất</p>
8.      <p>Đoạn văn thứ hai</p>
9.    </div>
10.   <h1>Phần 2</h1>
11.   <div id="container2"><p>Đoạn văn sẽ được thay thế</p></div>
12.   <script>
13.     var p3 = document.createElement("p");
14.     var p4 = document.createElement("p");
15.     var p5 = document.createElement("p");
16.     var txt3 = document.createTextNode("Đoạn văn thứ ba");
17.     var txt4 = document.createTextNode("Đoạn văn thứ tư");
18.     var txt5 = document.createTextNode("Đoạn văn thứ năm");
19.     p3.appendChild(txt3); p4.appendChild(txt4); p5.appendChild(txt5);
20.     var div1 = document.getElementById("container1");
21.     div1.appendChild(p4);
22.     div1.insertBefore(p3, p4);
23.     var div2 = document.getElementById("container2");
24.     var rp = document.querySelector("#container2 p");
25.     div2.replaceChild(p5, rp);
26.   </script>
27. </body></html>
```

4.4.6. Mở cửa sổ mới và tương tác giữa các đối tượng ở các cửa sổ khác nhau

Đối tượng *window* đại diện cho cửa sổ/tab của trình duyệt đang tải và hiển thị trang web. Nó là gốc của cây BOM như đã được trình bày trong Mục 4.3. Một điểm đáng lưu ý là đối tượng khung nội tuyến (*iframe*) có cửa sổ riêng của nó, có tên là *contentWindow*. *contentWindow* cũng là *window*, chỉ khác là phạm vi của nó trong một khung nội tuyến. Nói cách khác, *contentWindow* là gốc cây BOM của khung nội tuyến, *contentWindow.document* là gốc cây DOM của trang web bên trong (trang được tải và hiển thị trong khung nội tuyến). Mặt khác, *contentWindow*

là một nút con thuộc cây DOM của trang web bên ngoài (trang chứa khung nội tuyến). *contentWindow* có thuộc tính *parent* để tham chiếu đến đối tượng *window* của cửa sổ bên ngoài.

Thông qua *contentWindow*, từ trang web bên ngoài có thể truy cập đến các đối tượng tài liệu thuộc trang web bên trong. Ngược lại, thông qua đối tượng *parent*, từ trang web bên trong có thể truy cập đến các đối tượng tài liệu thuộc trang web bên ngoài. Trang *OuterPage.htm* trong ví dụ sau đây có chứa một khung nội tuyến tải trang *InnerPage.htm*. Mỗi trang có một đối tượng nhập chữ đều có định danh là *txt*. Mã JavaScript thực hiện đồng bộ giá trị giữa hai đối tượng nhập chữ này. Cụ thể, khi người dùng nhập vào một trong hai ô nhập chữ, nội dung nhập sẽ được cập nhật sang ô chữ còn lại.

```
1.  <!-- OuterPage.htm -->
2.  <!DOCTYPE html><html><head>
3.    <title>Outer Page</title>
4.    <meta charset="utf-8"/>
5.  </head><body>
6.    <input id="txt" type="text">
7.    <iframe id="ifr" src="InnerPage.htm"></iframe>
8.    <script>
9.      document.getElementById("txt").onkeyup = function() {
10.       var iwnd = document.getElementById("ifr").contentWindow;
11.       iwnd.document.getElementById("txt").value = this.value;
12.     };
13.   </script>
14. </body></html>
```

```
1.  <!-- InnerPage.htm -->
2.  <!DOCTYPE html><html><head>
3.    <title>Inner Page</title>
4.    <meta charset="utf-8"/>
5.  </head><body>
6.    <input id="txt" type="text">
7.    <script>
8.      document.getElementById("txt").onkeyup = function() {
9.        parent.document.getElementById("txt").value = this.value;
10.     };
11.   </script>
12. </body></html>
```

Một cửa sổ có thể mở ra một cửa sổ mới, sử dụng phương thức *window.open(url, …)*. Phương thức *open()* trả về tham chiếu của cửa sổ được mở. Ở chiều ngược lại, thuộc tính *opener* lưu tham chiếu đến cửa sổ đã mở ra cửa sổ hiện tại. Sử dụng hai tham chiếu vừa nêu có thể thực hiện các tương tác giữa các đối tượng tài liệu thuộc hai cửa sổ. Trang *OpeningPage.htm* trong ví dụ sau đây tải trang *OpenedPage.htm* ở một cửa sổ mới. Mỗi trang có một đối tượng nhập chữ đều có định danh là *txt*. Mã JavaScript thực hiện đồng bộ giá trị giữa hai đối tượng nhập chữ giữa hai trang. Cụ thể, khi người dùng nhập vào ô nhập chữ ở một trong hai trang, nội dung nhập sẽ được cập nhật sang ô chữ ở trang còn lại.

```
1. <!-- OpeningPage.htm -->
2. <!DOCTYPE html><html><head>
3.   <title>Opening Page</title>
4.   <meta charset="utf-8"/>
5. </head><body>
6.   <input id="txt" type="text">
7.   <script>
8.     var cwnd = window.open("OpenedPage.htm", "", "");
9.     document.getElementById("txt").onkeyup = function() {
10.      cwnd.document.getElementById("txt").value = this.value;
11.    };
12.  </script>
13. </body></html>
```

```
1. <!-- OpenedPage.htm -->
2. <!DOCTYPE html><html><head>
3.   <title>Opened Page</title>
4.   <meta charset="utf-8"/>
5. </head><body>
6.   <input id="txt" type="text">
7.   <script>
8.     document.getElementById("txt").onkeyup = function() {
9.       opener.document.getElementById("txt").value = this.value;
10.    };
11.  </script>
12. </body></html>
```

Tương tác giữa các đối tượng tài liệu thuộc hai cửa sổ khác nhau chỉ thực hiện được nếu hai trang web được tải vào hai cửa sổ thuộc cùng một ứng dụng và được truy cập theo cùng một lược đồ. Nói cách khác, URL của hai trang web phải có cùng lược đồ, tên miền, và số hiệu cổng. Điều kiện ràng buộc này được gọi là **Chính sách cùng nguồn gốc**, viết tắt là **SOP** (Same-origin policy). SOP là một khái niệm quan trọng trong mô hình an ninh ứng dụng web. Theo SOP, trình duyệt chỉ cho phép kịch bản ở trang này truy cập dữ liệu ở trang khác nếu hai trang có cùng nguồn gốc. SOP ngăn ngừa các kịch bản xấu từ một trang truy cập dữ liệu nhạy cảm từ trang khác thông qua truy cập DOM.

4.4.7. Hộp thoại, in ấn

Phương thức *window.alert(msg)* hiển thị một thông báo có nội dung là *msg*. Phương thức *window.confirm(msg)* hiển thị một hộp thoại hỏi xác nhận có nội dung là *msg* với hai nút có nhãn là "OK" và "Cancel". Nếu người dùng bấm nút "OK", hộp thoại được đóng và trả về *true*. Ngược lại, nếu người dùng bấm nút "Cancel" hoặc đóng hộp thoại theo cách khác, hộp thoại sẽ trả về *false*. Phương thức *window.print()* cho phép in nội dung được hiển thị trên trang web.

4.5. AJAX

Cửa sổ (*window* của tab trình duyệt hay *contentWindow* của *<iframe>*) là phương tiện để tải và hiển thị trang web. Tại một thời điểm, mỗi cửa sổ chỉ có thể tải nhiều nhất một trang web và quản lý duy nhất một cây DOM. Khi tải trang mới, cây DOM cũ bị xóa đi và được thay thế bởi cây DOM mới. Mặt khác, tương tác giữa cửa sổ với trình phục vụ web được thực hiện một cách đồng bộ, nghĩa là mỗi khi cửa sổ gửi đi một yêu cầu HTTP, nó phải đợi cho đến khi nhận được đáp ứng HTTP thì mới làm công việc gì khác. Trong lúc đợi nhận đáp ứng HTTP, cửa sổ không làm gì và hoàn toàn không có tương tác với người dùng. Do vậy, nếu ứng dụng web chỉ sử dụng cửa sổ để tải các trang web và tài nguyên web thì hiệu quả sử dụng sẽ không cao, đồng thời không thân thiện với người dùng. Hiệu quả sử dụng không cao là bởi trang web sau có rất nhiều nội dung trùng lặp với trang web trước (ví dụ, cấu trúc, hệ thống thực đơn, các bài liên quan, …) nhưng cửa sổ không "sử dụng lại". Thay vào đó, cửa sổ xóa bỏ toàn bộ nội dung của trang trước để thay bằng nội dung của trang sau. Điều đó dẫn đến không những thời gian xử lý ở cả hai phía (khách và phục vụ) lớn mà dung lượng truyền tải giữa hai bên cũng cao. Mặt khác, tính thân thiện không cao vì người dùng phải tương tác với ứng dụng theo kiểu "dừng-và-đợi" mỗi khi cửa sổ tải trang mới.

Nhằm khắc phục những hạn chế khi sử dụng cửa sổ như đã được trình bày ở trên, các ứng dụng web ngày nay khai thác triệt để kỹ thuật AJAX (Asynchronous JavaScript and XML). Về mặt công nghệ, AJAX là đối tượng JavaScript hoạt động bên trong cửa sổ, giao tiếp với trình phục vụ ở chế độ nền (background) thay cho cửa sổ trong lúc cửa sổ đang thực hiện một cách liên tục các công việc quản lý trang web và tương tác với người dùng. Sử dụng AJAX, cửa sổ không phải tải lại trang. Mỗi khi đối tượng AJAX nhận được dữ liệu mới, cửa sổ không xóa bỏ cây DOM của nó mà chỉ sửa đổi cây DOM cho phù hợp với nội dung mới. Làm như vậy, cửa sổ có thể "sử dụng lại" nội dung đã có và không phải tải lại. Mặt khác, cửa sổ có thể tương tác với người dùng một cách liên tục, không phải "dừng-và-đợi".

Trước đây, dữ liệu trong đáp ứng HTTP khi sử dụng AJAX thường là XML (eXtensible Markup Language). XML được trình phục vụ gửi đến đối tượng AJAX và cửa sổ sử dụng JavaScript để phân tích nội dung XML. AJAX cho phép cửa sổ cập nhật trang web một cách không đồng bộ, nghĩa là cửa sổ không phải đợi nhận được đáp ứng HTTP mới được làm công việc khác. Thuật ngữ AJAX bắt nguồn từ những đặc điểm kỹ thuật vừa được phân tích.

Đối tượng AJAX, còn gọi là mô-tơ AJAX (AJAX engine), có tên lớp khác nhau trong các trình duyệt. Đa số các trình duyệt hiện đại sử dụng tên lớp *XMLHttpRequest* cho đối tượng AJAX. Tuy nhiên, một số trình duyệt (ví dụ IE) sử dụng *ActiveXObject*. Do vậy, để khai báo đối tượng AJAX chạy trên mọi trình

duyệt, các tên lớp khác nhau phải được kết hợp. Hàm khai báo đối tượng AJAX có thể được cài đặt như sau:

```
1. /* File: xmlhttp.js */
2. function getXmlHttpObject()  {
3.   var xmlhttp = null;
4.   try { xmlhttp = new XMLHttpRequest(); }
5.   catch (e) {
6.     try  { xmlhttp = new ActiveXObject("Msxml2.XMLHTTP"); }
7.     catch (e) {
8.       try { xmlhttp = new ActiveXObject("Microsoft.XMLHTTP"); }
9.       catch (e)  { console.log("Trình duyệt không hỗ trợ AJAX!"); }
10.    }
11.  }
12.  return xmlhttp;
13. }
```

Để gửi yêu cầu tới trình phục vụ, đối tượng AJAX sử dụng kết hợp hai phương thức *open()* và *send()*. Ngoài ra, đối tượng AJAX cung cấp phương thức *setRequestHeader()* để thiết lập các tiêu đề cho yêu cầu. Phương thức *open()* có ba tham số bắt buộc lần lượt là động từ (GET, POST, ...), URL, và tính đồng bộ. Có thể cung cấp thêm tên sử dụng và mật khẩu cho *open()* ở các tham số thứ tư và thứ năm. Nếu gửi yêu cầu POST, tiêu đề *Content-Type* cần phải được đặt giá trị là "*application/x-www-form-urlencoded*", đồng thời các tham số POST được truyền cho *send()*. Các lệnh gửi yêu cầu GET có dạng:

xmlhttp.open("GET", path?querystring, true|false);
xmlhttp.send(null);

và các lệnh gửi yêu cầu POST có dạng:

xmlhttp.open("POST", url, true|false);
xmlhttp.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
xmlHttp.send(params);

Đối tượng AJAX có thuộc tính *readyState* cho biết trạng thái xử lý yêu cầu. Giá trị của *readyState* và ý nghĩa của chúng được mô tả như sau:

Giá trị	Ý nghĩa
0	Chưa thiết lập yêu cầu.
1	Đã thiết lập kết nối đến trình phục vụ.
2	Đã gửi yêu cầu nhưng chưa có hồi âm.
3	Đang nhận đáp ứng.
4	Đã nhận xong đáp ứng.

Đối tượng AJAX sẽ theo dõi trạng thái xử lý yêu cầu và cập nhật giá trị tương ứng cho thuộc tính *readyState*. Không những vậy, mỗi khi thuộc tính *readyState* được cập nhật, *onreadystatechange*, là một con trỏ sự kiện của đối tượng AJAX, cũng đồng thời được gọi. Khi nhận được đáp ứng, đối tượng AJAX cập nhật thuộc tính

status với mã trạng thái HTTP của đáp ứng. Hàm xử lý sự kiện *onreadystatechange* cần phân tích các giá trị của *readyState* và *status* để biết trạng thái đáp ứng. Ngoài ra, dữ liệu nhận được sẽ được đối tượng AJAX lưu vào thuộc tính *responseText* hoặc *responseXML* tùy thuộc vào kiểu MINE của đáp ứng là *text/html* hay *text/xml*, tương ứng. Do vậy, hàm xử lý sự kiện *onreadystatechange* cần đọc giá trị của một trong hai thuộc tính *responseText* hoặc *responseXML* để lấy dữ liệu.

Lưu ý, *onreadystatechange* cần phải được thiết lập trước khi đối tượng AJAX gửi yêu cầu. Cài đặt tổng quát cho xử lý AJAX có dạng như sau:

```
1. <script src="xmlhttp.js"></script>
2. <script>
3.   var xmlhttp = getXmlHttpObject();
4.   xmlhttp.onreadystatechange = function() {
5.     if (this.readyState == 0) {  …  }
6.     else if (this.readyState == 1) {  …  }
7.     else if (this.readyState == 2) {  …  }
8.     else if (this.readyState == 3) {  …  }
9.     else if (this.readyState == 4) {
10.      if (this.status == 200) {
11.        var data = this.responseText; // hoặc this.responseXML
12.        // Sử dụng Javascript cập nhật DOM theo data
13.      } else if (this.status == …) { … }
14.    }
15.  };
16.  xmlhttp.open(verb, url, true|false);
17.  xmlhttp.setRequestHeader(header, headerValue);
18.  xmlhttp.send(params|null);
19. </script>
```

Trang web ví dụ sau đây sử dụng AJAX để tải nội dung tệp *webpart.htm* rồi sử dụng JavaScript để đưa nội dung đó vào một đối tượng * *. Việc tải *webpart.htm* được thực hiện sau khi đã tải trang chính. Tệp *webpart.htm* trong trường hợp này chứa một phần mã HTML của trang web.

```
1. <!DOCTYPE html><html><head>
2.   <title>L.4.5</title>
3.   <meta charset="utf-8">
4. </head><body>
5.   <div id="div1"></div>
6.   <script type="text/javascript" src="xmlhttp.js"></script>
7.   <script>
8.     var xmlhttp = getXmlHttpObject();
9.     xmlhttp.onreadystatechange = function() {
10.      if (this.readyState == 4) {
11.        if (this.status == 200) {
12.         document.getElementById("div1").innerHTML = this.responseText;
13.        }
14.      }
15.    };
16.    xmlhttp.open("GET", "webpart.htm", true);
17.    xmlhttp.send(null);
18.  </script>
19. </body></html>
```

4.6. JSON

Yêu cầu và đáp ứng HTTP đều có định dạng văn bản. Dữ liệu được trao đổi giữa trình khách và trình phục vụ web, do đó, cũng chỉ là văn bản. JSON (JavaScript Object Notation) là cú pháp được sử dụng rộng rãi trong lưu trữ cũng như trao đổi dữ liệu bởi ứng dụng web. Ưu điểm của JSON là dễ hiểu, độc lập với ngôn ngữ và nhẹ (biểu diễn có dung lượng thấp). Hơn nữa, JSON còn có cú pháp tương tự với JavaScript, có thể chuyển đổi đối tượng JavaScript thành JSON và ngược lại một cách đơn giản, có thể dễ dàng gửi/nhận dữ liệu JSON đến/từ trình phục vụ. Theo đó, ứng dụng web có thể làm việc với dữ liệu như những đối tượng JavaScript.

Cú pháp JSON được kế thừa và là tập con của cú pháp JavaScript. Dữ liệu được biểu diễn bằng các giá-trị. Giá-trị có thể là số, xâu, *null*, boolean, mảng và đối tượng JSON. Mảng là danh sách các giá-trị được phân cách bởi dấu phẩy và đặt giữa các dấu mở và đóng ngoặc vuông ([và]). Đối tượng có cú pháp tựa đối tượng JavaScript chỉ khác là tên thuộc tính phải được đặt giữa hai dấu nháy kép. Lưu ý, khác với JavaScript, JSON không có dữ liệu kiểu ngày tháng, hàm, *undefined*. Khi chuyển một đối tượng JavaScript thành đối tượng JSON, dữ liệu ngày tháng, *undefined* sẽ được chuyển thành xâu, hàm sẽ bị bỏ qua. Cũng lưu ý, mảng có thể chứa đối tượng và đối tượng có thể chứa mảng, mảng và đối tượng có thể lồng nhau; về chức năng ngôn ngữ, mảng và đối tượng cũng chỉ là giá-trị.

Ví dụ, dữ liệu JSON sau đây cung cấp thông tin về xe riêng của các nhân viên. Mỗi nhân viên có thông tin về tên, tuổi, tên và mẫu các xe riêng của họ. Dữ liệu là một danh sách bao gồm ba nhân viên, mỗi nhân viên là một đối tượng JSON với ba thuộc tính "*name*", "*age*" và "*cars*". Giá trị của "*cars*" lại là danh sách các đối tượng xe. Đối tượng xe có các thuộc tính "*name*" và "*models*" với giá trị của "*models*" là một danh sách các xâu. Dễ thấy, dữ liệu JSON tự mô tả nó, biểu diễn JSON rất ngắn gọn và dễ hiểu.

```
1. [
2.   {
3.     "name": "John",
4.     "age": 30,
5.     "cars": [
6.       { "name": "Ford", "models": [ "Fiesta", "Focus", "Mustang" ] },
7.       { "name": "BMW",  "models": [ "320", "X3" ] }
8.     ]
9.   },
10.  {
11.    "name": "Maria",
12.    "age": 25,
13.    "cars": [
14.      { "name": "Fiat", "models":[ "500", "Panda" ] }
15.    ]
16.  },
17.  {
18.    "name": "David",
19.    "age": 40,
```

```
20.     "cars": [
21.       { "name": "Ford", "models": [ "Fiesta", "Focus", "Mustang" ] },
22.       { "name": "BMW",  "models": [ "320", "X3", "X5" ] },
23.       { "name": "Fiat", "models": [ "500", "Panda" ] }
24.     ]
25.   }
26. ]
```

Khi nhận được dữ liệu JSON từ trình phục vụ, trình khách sử dụng JavaScript để đọc dữ liệu và cập nhật cây DOM cho phù hợp với dữ liệu. Mặc dù có thể thao tác trực tiếp trên các đối tượng hay mảng JSON, nhưng chuyển đổi đối tượng hay mảng JSON thành đối tượng JavaScript sẽ thuận tiện hơn. Phương thức *JSON.parse(jsonData)* cung cấp một API chuyển đổi hiệu quả. Đầu vào của phương thức này là dữ liệu JSON và kết quả trả về của nó là một đối tượng JavaScript. Trang web ví dụ sau đây sử dụng AJAX để tải dữ liệu JSON (về xe riêng của các nhân viên như được mô tả ở trên), chuyển đổi thành đối tượng JavaScript, rồi sử dụng JavaScript để tạo các đối tượng DOM *<tr>*, *<td>* và thêm vào thân của đối tượng bảng có định danh là *tbl1*. Kết quả, trang web hiển thị thông tin về xe của các nhân viên dưới dạng bảng.


```
1. <!DOCTYPE html><html><head>
2.   <title>L.4.6</title>
3.   <meta charset="utf-8">
4. </head><body>
5.   <table id="tbl1">
6.     <thead>
7.       <th>Họ và tên</th>
8.       <th>Tuổi</th>
9.       <th>Số lượng/Tên-mẫu xe</th>
10.    </thead>
11.    <tbody></tbody>
12.  </table>
13.  <script type="text/javascript" src="xmlhttp.js"></script>
14.  <script>
15.    var xmlhttp = getXmlHttpObject();
16.    xmlhttp.onreadystatechange = function() {
17.      if (this.readyState == 4) {
18.        if (this.status == 200) {
19.          var obj = JSON.parse(this.responseText);
20.          for (var i = 0; i < obj.length; i++) {
21.            var r = document.createElement("tr");
22.            var c1 = document.createElement("td");
23.            var c2 = document.createElement("td");
24.            var c3 = document.createElement("td");
25.            c1.innerHTML = obj[i].name;
26.            c2.innerHTML = obj[i].age;
27.            c3.innerHTML = obj[i].cars.length;
28.            for (var j = 0; j < obj[i].cars.length; j++)
29.              c3.innerHTML += obj[i].cars[j].name + obj[i].cars[j].models;
30.            r.appendChild(c1);
31.            r.appendChild(c2);
32.            r.appendChild(c3);
33.            document.querySelector("#tbl1 tbody").appendChild(r);
34.          }
35.        }
```

```
36.       }
37.     };
38.     xmlhttp.open("GET", "json.txt", true);
39.     xmlhttp.send(null);
40.   </script>
41. </body></html>
```

Dữ liệu được trình khách quản lý trong các đối tượng JavaScript. Khi cần gửi dữ liệu cho trình phục vụ, trình khách chuyển đổi đối tượng JavaScript thành đối tượng JSON và gửi dữ liệu JSON cho trình phục vụ. Phương thức *JSON.stringify(obj)* cung cấp một API hiệu quả cho việc chuyển đổi này. Đầu vào của phương thức này là đối tượng JavaScript và kết quả trả về của nó là một đối tượng hay dữ liệu JSON.

Việc tiếp nhận, xử lý hay tạo ra dữ liệu JSON phía trình phục vụ sẽ được trình bày trong Chương 6.

4.7. VẤN ĐỀ CỦA TRÌNH DUYỆT

Trong thực tế, một trang web (cùng mã nguồn) có thể được trình diễn không giống nhau bởi các trình duyệt. Điều này cũng dễ hiểu vì trình duyệt là trình thông dịch và mỗi trình duyệt có thể phân tích mã nguồn cũng như thực thi trang web theo thuật toán riêng của nó. Ví dụ, đối tượng *<input type="date">* hiện tại không được hỗ trợ bởi Firefox trong khi được hỗ trợ bởi Chrome. Ví dụ khác về CSS, thuộc tính tạo hiệu ứng tịnh tiến 2D có tên là *transform*, trong khi IE9 cài đặt thuộc tính là *–ms-transform*, Opera cài đặt thuộc tính là *–o-transform*. JavaScript cũng có vấn đề tương tự, ví dụ trình duyệt Firefox sử dụng phương thức *addEventListener()* để gán hàm xử lý sự kiện cho đối tượng tài liệu, trong khi trình duyệt IE thực hiện công việc này bằng phương thức *attachEvent()*.

Tất nhiên, các hãng phát triển trình duyệt phải tuân thủ các đặc tả (RFC hoặc Recommendation) của W3C để trình duyệt có thể dùng được. Sự khác biệt giữa các trình duyệt là không đáng kể so với phần giống nhau giữa chúng. Tuy nhiên, chỉ cần một sự khác biệt cũng dẫn đến kết quả trình diễn trang web khác.

Mong muốn của người phát triển ứng dụng web là ứng dụng có thể chạy trên mọi trình duyệt (cross-browser). Để đạt được như vậy, giải pháp tổng quát là người lập trình phải nắm rõ khả năng hỗ trợ HTML, CSS, JavaScript, DOM, và BOM của các trình duyệt, từ đó viết mã chung chạy được trên mọi trình duyệt hoặc tùy vào trình duyệt mà viết mã khác nhau. Mỗi khi sử dụng một thẻ HTML, thuộc tính CSS hay phương thức JavaScript, lập trình viên cần biết rõ thẻ/thuộc tính/phương thức đó được hỗ trợ bởi những trình duyệt nào. Đối tượng *window* có thuộc tính *clientInformation* cho biết thông tin về trình duyệt và hệ điều hành phía khách. Thông tin về trình duyệt và hệ điều hành phía khách cũng được gửi đến trình phục vụ trong các yêu cầu HTTP. Căn cứ những thông tin này, ứng dụng có thể tạo mã khác nhau cho phù hợp với trình duyệt.

Một vấn đề nổi bật liên quan trình duyệt và thiết bị là giao diện của trang web cần phải thích ứng với kích thước khung nhìn của trình duyệt và kích thước màn hình của thiết bị. Ở những khung nhìn có kích thước khác nhau, giao diện tự điều chỉnh sao cho người dùng dễ quan sát và tương tác nhất. Những trang web có tính năng như vậy được gọi là có thiết kế đáp ứng (responsive design).

Chạy được trên mọi trình duyệt và có thiết kế đáp ứng là những tính năng cần có của ứng dụng web hiện đại vì người dùng ngày nay không chỉ sử dụng máy tính mà còn sử dụng nhiều thiết bị khác như điện thoại thông minh, PDA, … để truy cập web.

4.8. CẬP NHẬT KIẾN THỨC VỀ JAVASCRIPT

Đặc tả của JavaScript là ECMAScript (viết tắt là ES) được ECMA International chuẩn hóa trong ECMA-262. Các phiên bản của ECMAScript được đánh số thứ tự hoặc đặt tên theo năm phát hành. Ví dụ, ES6 tương đương ECMAScript 2015, là phiên bản được phát hành năm 2015. Tháng 6/2017, ECMA International đã công bố ES8. Lưu ý rằng phiên bản JavaScript không đồng nhất với phiên bản của ECMAScript. Ví dụ, JavaScript 1.3 (năm 1998) tương đương ES2, trong khi JavaScript 1.8 (năm 2008) tương đương ES3. Thực tế này dễ hiểu vì JavaScript là một cài đặt cụ thể của ECMAScript, trong khi ECMAScript là đặc tả. Ngoài JavaScript, nhiều ngôn ngữ kịch bản khác như JScript, ActionScript, SpiderMonkey cũng là các cài đặt cụ thể của ECMAScript. Mặt khác, một vài phiên bản mới của ECMAScript có thể chưa được hiện thực hóa trong các cài đặt JavaScript, do vậy ECMAScript phiên bản mới được coi là phiên bản tiếp theo của JavaScript. Để sử dụng ECMAScript phiên bản mới nhất trong các ứng dụng, lập trình viên có thể sử dụng công cụ chuyển đổi, ví dụ Babel tại *https://babeljs.io/*, để chuyển mã nguồn ES phiên bản mới về JavaScript mà trình duyệt hiểu được. Babel là một trong số các công cụ tiền xử lý JavaScript đang được sử dụng rộng rãi.

Bài tập

1. Tạo trang web để nhập thông tin cá nhân như sơ yếu lý lịch, sử dụng JavaScript để kiểm tra dữ liệu được nhập có hợp lệ hay không.
2. Tạo trang web có bảng được khai báo bằng HTML. Sử dụng JavaScript để cho phép người dùng kích chuột vào ô tiêu đề của bảng và sắp xếp dữ liệu trong bảng theo cột có ô tiêu đề được kích.

Đọc thêm

1. Ved Antani, Stoyan Stefanov, "Object-Oriented JavaScript, 3rd Edition", Packt Publishing, 2017.

2. Gilad Tsur Mayer, "JavaScript: JavaScript Awesomeness Book", Amazon Digital Services LLC, 2016.
3. ECMA International, "ECMA-262, ECMAScript® 2017 Language Specification", 2017.