|  |  |
| --- | --- |
| MỤC LỤC | |
| |  | | --- | | [CSS](#_Toc475120636)   * [CÚ PHÁP](#_Toc475120637) * [THỨ TỰ KHAI BÁO](#_Toc475120638) * [KHÔNG SỬ DỤNG @IMPORT](#_Toc475120639) * [VỊ TRÍ ĐẶT MEDIA QUERY](#_Toc475120640) * [THUỘC TÍNH TIỀN TỐ](#_Toc475120641) * [KHAI BÁO CSS ĐƠN](#_Toc475120642) * [KÍ HIỆU VIẾT TẮT](#_Toc475120643) * [QUY TẮC XẾP CHỒNG TRONG LESS VÀ SASS](#_Toc475120644) * [CÁC PHÉP TÍNH TRONG LESS VÀ SASS](#_Toc475120645) * [GHI CHÚ (COMMENTS)](#_Toc475120646) * [TÊN CLASS](#_Toc475120647) * [BỘ CHỌN](#_Toc475120648) * [CƠ CẤU TỔ CHỨC](#_Toc475120649) * [TÙY CHỌN CHO CHƯƠNG TRÌNH VIẾT CODE](#_Toc475120650) | | | |
| CSS | |
| Cú pháp  * Sử dụng tabs ngắn (2 khoảng trắng) - đó là cách duy nhất để đảm bảo code được dịch giống nhau trong mọi môi trường. * Khi nhóm các bộ chọn (selectors), hãy giữ mỗi bộ chọn nằm trên một dòng. * Thêm một khoảng trắng trước một khối của thuộc tính để dễ đọc. * Đặt dấu đóng ngoặc nhọn của một khối thuộc tính ở một dòng mới. * Thêm một khoảng trắng phía sau dấu: của mỗi thuộc tính. * Với mỗi định nghĩa chỉ nên nằm trên riêng một dòng sẽ cho báo lỗi được chính xác hơn. * Luôn sử dụng dấu chấm phẩy để kết thúc tất các định nghĩa. Dấu chấm phẩy ở định nghĩa cuối cùng là không bắt buộc nhưng lỗi của bạn có thể xảy ra nhiều lỗi nếu không có nó. * Thêm khoảng trắng phía sau dấu phẩy ở mỗi định nghĩa có nó (ví dụ box-shadow). * Không thêm khoảng trắng vào phía sau dấu phẩy trong giá trị của rgb(), rgba(), hsl(), hsla(), hoặc rect().Nó giúp phân biệt giữa các giá trị màu sắc (dấu phẩy, không khoảng trắng) và các giả trị thuộc tính (có dấu phẩy cùng khoảng cách). * Không cần phải thêm tiền tố với các giá trị hoặc màu sắc bắt đầu bằng số 0 (ví dụ: .5 thay vì 0.5 và -.5px thay vìo -0.5px). * Sử dụng chữ thường cho giá trị Hex của màu sắc, ví dụ #fff. Chữ thường sẽ giúp việc dịch code được dễ dàng khi gặp những hình khối phức tạp. * Sử dụng kiểu hiển thị giá trị màu sắc Hex theo kiểu rút ngọn nếu có thể, ví dụ #fffthay vì#ffffff. * Sử dụng dấu ngoặc kép để khai báo thành phần giá trị trong bộ chọn, ví dụ input[type="text"]. [Nó chỉ bắt buộc trong một số trường hợp nhất định](http://mathiasbynens.be/notes/unquoted-attribute-values#css), nhưng sẽ mang tính nhất quán nếu sử dụng cho toàn bộ. * Tránh thêm đơn vị cho giá trị bằng 0, ví dụ margin: 0; thay vì margin: 0px;. * Nếu bạn có thắc mắc về các vấn đề trên? Hãy đọc [Cú pháp viết Cascading Style Sheets](http://en.wikipedia.org/wiki/Cascading_Style_Sheets#Syntax) trên Wikipedia. | /\* Bad CSS \*/  .selector, .selector-secondary, .selector[type=text] {  padding:15px;  margin:0px 0px 15px;  background-color:rgba(0, 0, 0, 0.5);  box-shadow:0px 1px 2px #CCC,inset 0 1px 0 #FFFFFF  }  /\* Good CSS \*/  .selector,  .selector-secondary,  .selector[type="text"] {  padding: 15px;  margin-bottom: 15px;  background-color: rgba(0,0,0,.5);  box-shadow: 0 1px 2px #ccc, inset 0 1px 0 #fff;  } |
| Thứ tự khai báo Những thuộc tính có đặc tính giống nhau nên được sắp xếp theo thứ tự sau:   1. Định vị trí (Positioning) 2. Kiểu hình khối (Box model) 3. Kiểu kí tự (Typographic) 4. Hình ảnh (Visual)   Định vị trí đầu tiên vì nó có thể loại bỏ phần tử trong một layout bình thường. Kiểu hình khối được định nghĩa tiếp theo để có thể xác định được khích thước của mỗi thành phần.  Các thành phần còn lại hãy đặt theo từng thành phần tương ứng những phần mà không ảnh hưởng đến giá trị đã đặt trước đó sẽ đặt ở cuối cùng.  Cho việc hoàn thành danh sách các thuộc tính theo thứ tự, xin hãy tìm hiểu thêm tại [Recess](http://twitter.github.com/recess). | .declaration-order {  /\* Positioning \*/  position: absolute;  top: 0;  right: 0;  bottom: 0;  left: 0;  z-index: 100;  /\* Box-model \*/  display: block;  float: right;  width: 100px;  height: 100px;  /\* Typography \*/  font: normal 13px "Helvetica Neue", sans-serif;  line-height: 1.5;  color: #333;  text-align: center;  /\* Visual \*/  background-color: #f5f5f5;  border: 1px solid #e5e5e5;  border-radius: 3px;  /\* Misc \*/  opacity: 1;  } |
| Không sử dụng @import So với các <link>, @import sẽ chậm hơn, nó sẽ cần nhiều yêu cầu từ trang hơn và không dự tính được hết các lỗi có thể xảy ra. Hãy dùng một số cách thay thế sau thay vì sử dụng @import:   * Sử dụng nhiều <link> * Sử dụng các trình phiên dịch như SASS & LESS để phiên dịch CSS vào trong một file duy nhất * Kết nối các files CSS của bạn bằng các tính năng được môi trường cung cấp sẵn nhu Rail, Jekyll và các môi trường khác   Để thông tin chi tiết hơn, xin học [bài viết của Steve Souders](http://www.stevesouders.com/blog/2009/04/09/dont-use-import/). | <!-- Use link elements -->  <link rel="stylesheet" href="core.css">  <!-- Avoid @imports -->  <style>  @import url("more.css");  </style> |
| Vị trí đặt Media query Hãy đặt Media query gần với bộ chọn có liên quan khi có thể. Đừng gom tất cả các quy tắc trong một khu vực riêng biệt như phần cuối trang. Làm như vậy sẽ giúp cho nhớ lại những truy vấn dễ dàng hơn trong tương lai. Đây là một ví dụ. | .element { ... }  .element-avatar { ... }  .element-selected { ... }  @media (min-width: 480px) {  .element { ...}  .element-avatar { ... }  .element-selected { ... }  } |
| Thuộc tính tiền tố Khi sử dụng các thuộc tính tiền tố (Prefixed properties), hãy thụt đầu dòng vào dể các giá trị có thể thẳng hàng với nhau giúp dễ dàng để thay đổi giá trị.  Trong Textmate, chọn **Text → Edit Each Line in Selection** (⌃⌘A). Trong Sublime Text 2, chọn **Selection → Add Previous Line** (⌃⇧↑) và **Selection → Add Next Line** (⌃⇧↓). | /\* Prefixed properties \*/  .selector {  -webkit-box-shadow: 0 1px 2px rgba(0,0,0,.15);  box-shadow: 0 1px 2px rgba(0,0,0,.15);  } |
| Khai báo CSS đơn Trong trường hợp **chỉ có 1 định nghĩa CSS**, hãy loại bỏ những ngắt dòng để có thể dễ chỉnh sửa hơn. Bất kì CSS nào sử dụng nhiều định nghĩa đều phải cho xuông dòng.  Đây là một trong yếu tố quan trọng để phát hiện ra lỗi—ví dụ một CSS đang có báo lỗi ở Line 183. Với định nghĩa nằm ở 1 dòng, bạn sẽ không bị mất nó. Với nhiều định nghĩa nằm riêng biệt trên mỗi dòng bạn sẽ dễ dàng phân biệt được. | /\* Single declarations on one line \*/  .span1 { width: 60px; }  .span2 { width: 140px; }  .span3 { width: 220px; }  /\* Multiple declarations, one per line \*/  .sprite {  display: inline-block;  width: 16px;  height: 15px;  background-image: url(../img/sprite.png);  }  .icon { background-position: 0 0; }  .icon-home { background-position: 0 -20px; }  .icon-account { background-position: 0 -40px; } |
| Kí hiệu viết tắt Hãy hạn chế các kí hiệu viết tắt, thay vào đó hãy định nghĩa rõ ràng các giá trị. Các thuộc tính viết tắt thường được sử dụng nhiều bao gồm:   * padding * margin * font * background * border * border-radius   Thông thường bạn không cần phải điền hết các thuộc tính của định nghĩa viết tắt. Ví dụ : HTML heading chỉ cần setting cho top và bottom margin, đến lúc cần, chỉ cần viết lại 2 thuộc tính đó. Sử dụng đinh nghĩa viết tắt quá nhiều sẽ dẫn dến các định nghĩa bị lặp lại quá nhiều và xảy ra lỗi ngoài ý muốn.  Tài liệu Mozilla Developer Network có một bài viết hay về [shorthand properties](https://developer.mozilla.org/en-US/docs/Web/CSS/Shorthand_properties) cho người không quen với việc viết tắt này. | /\* Bad example \*/  .element {  margin: 0 0 10px;  background: red;  background: url("image.jpg");  border-radius: 3px 3px 0 0;  }  /\* Good example \*/  .element {  margin-bottom: 10px;  background-color: red;  background-image: url("image.jpg");  border-top-left-radius: 3px;  border-top-right-radius: 3px;  } |
| Quy tắc xếp chồng trong Less và Sass Hạn chế phần xếp chồng không cần thiết. Bạn có thể xếp chồng các bộ chọn, không có nghĩa là bạn luôn xếp chồng mọi thứ lại với nhau. Hãy xem xét phạm vi yếu tố có thể tác động tới phần cha, và các yếu tố được lồng nhau.  Đọc thêm:   * [Nesting in Sass and Less](http://markdotto.com/2015/07/20/css-nesting/) | // Without nesting  .table > thead > tr > th { … }  .table > thead > tr > td { … }  // With nesting  .table > thead > tr {  > th { … }  > td { … }  } |
| Các phép tính trong Less và Sass Để giúp dễ đọc hơn, hãy đặt các phép tính nằm ở trong ngặc đơn và thêm khoảng trắng còa giữa các giá trị, biến và các phép tính. | // Bad example  .element {  margin: 10px 0 @variable\*2 10px;  }  // Good example  .element {  margin: 10px 0 (@variable \* 2) 10px;  } |
| Ghi chú (Comments) Code được viết và duy trì giữa nhiều người. Vì vậy hãy đảm bảo rằng các mô tả, ghi chú của bạn người khác cũng có thể đọc được. Một ghi chú code tốt là một ghi chú có bối cảnh và mục đích. Không nên ghi chú đơn giản bằng một phần đại diện hoặc tên lớp.  Hãy đảm bảo rằng các ghi chú là một câu cú hoàn chỉnh, cũng như sử dụng các cụm từ ngắn gọn dễ hiểu cho các ghi chú chung. | /\* Bad example \*/  /\* Modal header \*/  .modal-header {  ...  }  /\* Good example \*/  /\* Wrapping element for .modal-title and .modal-close \*/  .modal-header {  ...  } |
| Tên Class  * Sử dụng chữ viết thường và dấu gạch ngắn (không sử dụng dấu gạch dài và chữ hoa).Dấu gạch ngang chỉ sử dụng dùng dể nối từ hoặc các chức năng có liên quan (ví dụ .btn và .btn-danger). * Tránh tự ý viết tắt tùy tiện và quá nhiều. .btn sẽ tôt cho button, tuy nhiên .s thì không có nghĩa. * Cố gắng để tên Class ngắn gọn. * Sử dụng tên có nghĩa, có mục đích. * Sử dụng tiền tố dựa trên lớp cha gần nhất. * Sử dụng classes .js-\* để phân biệt (không sử dụng style), để giúp cho các class này không sử dụng trong file CSS.   Các quy tắc trên cũng rất hữu dụng với các tên biến khi bạn sử dụng SASS và LESS. | /\* Bad example \*/  .t { ... }  .red { ... }  .header { ... }  /\* Good example \*/  .tweet { ... }  .important { ... }  .tweet-header { ... } |
| Bộ chọn  * Sử dụng Class cho những tag để tối ưu hiệu suất dịch code. * Hạn chế chọn phần tử kiểu chọn theo thuộc tính (attribute) (ví dụ [class^="..."]) . Hiệu suất trình duyệt sẽ bị ảnh hưởng. * Giữ cho tên Class ngắn và gọi phần cha ít. Cố gắng gọi tối đa 3 Class. * Gọi tên class cha **chỉ khi** nào cần thiết (ví dụ khi không cần tạo css tùy chỉnh).   Đọc thêm:   * [Scope CSS classes with prefixes](http://markdotto.com/2012/02/16/scope-css-classes-with-prefixes/) * [Stop the cascade](http://markdotto.com/2012/03/02/stop-the-cascade/) | /\* Bad example \*/  span { ... }  .page-container #stream .stream-item .tweet .tweet-header .username { ... }  .avatar { ... }  /\* Good example \*/  .avatar { ... }  .tweet-header .username { ... }  .tweet .avatar { ... } |
| Cơ cấu tổ chức  * Phân vùng code thành các phần riêng biệt. * Xây dựng cấu trúc chú thích hợp lý và thống nhất * Sử dụng nhiều khoảng trống lớn để phân cách cho những phần code lớn để có thể dễ dàng tìm kiếm khi làm việc với một file lớn. * Khi sử dụng nhiều file CSS, hãy phân vùng chúng theo từng thành phần thay vì phân theo các trang. Vì các trang có thể thay đổi thành phần và các thành phần cũng có thể di chuyển theo yêu cầu. | /\*  \* Component section heading  \*/  .element { ... }  /\*  \* Component section heading  \*  \* Sometimes you need to include optional context for the entire component. Do that up here if it's important enough.  \*/  .element { ... }  /\* Contextual sub-component or modifer \*/  .element-heading { ... } |
| Tùy chọn cho chương trình viết code Điều chỉnh chương trình viết code của bạn theo các thông số dưới đây để tránh xảy ra lỗi với code:   * Sử dụng tab ngắn (soft-tabs) là 2 khoảng trắng. * Tự động xóa khoảng trắng thừa khi save. * Đặt mã kí tự là UTF-8. * Thêm một dòng mới vào cuối file.   Hãy tham khảo và áp dụng những chỉnh sửa trên vào project của bạn bằng file .editorconfig. Ví dụ, hãy tham khảo file trên [Bootstrap](https://github.com/twbs/bootstrap/blob/master/.editorconfig). Tìm hiểu thêm về [EditorConfig](http://editorconfig.org/). | |