# CSS

Internal: sử dụng 1 cặp thẻ style trong file html

External: dùng file CSS bên ngoài link vào html

Inline: CSS ngay trong thẻ

ID (#) : Identifi định danh duy nhất, chỉ được xuất hiện 1 lần

Class (.) : Lớp được xuất hiện nhiều lần

Độ ưu tiên trong CSS:

1. Intenal, External : không có độ ưu tiên, viết sau thì đươc chọn
2. Inline – 1000
3. #id – 100
4. .class – 10
5. Tag – 1
6. Equal specificity (trùng tên element, mới hơn thì được chọn)
7. Universal selector anh inherited : phổ cập \* , 0đ

# QUY CHUẨN BEM

BEM là viết tắt của Block, Element, Modifier. Đây là một quy chuẩn đặt tên lớp CSS cho các thành phần của giao diện người dùng (UI) trên trang web hoặc ứng dụng web. BEM được phát triển bởi Yandex, một công ty tìm kiếm và dịch vụ trực tuyến của Nga.

BEM giúp định danh rõ ràng và cụ thể các thành phần của giao diện người dùng, đồng thời giúp tránh xung đột và đảm bảo tính tái sử dụng của mã CSS. Quy chuẩn BEM bao gồm các quy tắc sau:

* Block: Đại diện cho một phần tử lớn trên trang web. Các block thường là các thành phần độc lập, có thể được sử dụng ở nhiều vị trí trên trang. Ví dụ: header, menu, footer.
* Element: Là thành phần con của block, tạo thành các phần tử nhỏ hơn bên trong block. Các element được xác định bằng cách sử dụng tên block, theo sau là dấu gạch dưới và tên element. Ví dụ: menu\_\_item, header\_\_logo.
* Modifier: Là các biến thể hoặc tình huống đặc biệt của block hoặc element. Các modifier giúp kiểm soát trạng thái, kích thước, màu sắc hoặc phong cách của block hoặc element. Ví dụ: menu\_\_item--active, button--disabled.

Việc đặt tên cho các lớp CSS theo quy chuẩn BEM sẽ giúp code trở nên dễ hiểu và dễ bảo trì hơn, đồng thời giúp tránh xung đột với các lớp khác trên trang.

# BEM là gì? Quy ước đặt tên CSS theo tiêu chuẩn của BEM

BEM là một thỏa thuận về việc đặt tên cho CSS, được xem là tiêu chuẩn cơ bản cho những người làm việc trong nhóm. BEM phù hợp với các trang web sản xuất, phục vụ cho khách hàng cuối cùng thay vì các mẫu. Vậy BEM là gì và làm thế nào để đặt tên CSS theo tiêu chuẩn BEM một cách chính xác nhất?

**BEM là gì?**

BEM là viết tắt của cụm **Block – Element – Modifier**, là tiêu chuẩn quy ước đặt tên cho các tên lớp CSS. BEM giúp việc lập trình Front-end trở nên dễ hiểu, dễ đọc hơn, dễ làm việc và dễ mở rộng cũng như bảo trì khi làm việc với CSS.

Việc đặt tên theo tiêu chuẩn giúp các lập trình viên frontend hiểu được đoạn code đó có nghĩa là gì, nó thực hiện nhiệm vụ gì? Từ đó những lập tình viên khác khi đọc code của họ cũng hiểu được bạn đang làm gì, từ đó dễ dàng phân tích, thay đổi và quản lý.

**Tại sao nên dùng BEM?**

Đối với những dự án nhỏ thì BEM có thể chưa cần đến, tuy nhiên các dự án lớn và làm việc nhóm thì BEM rất quan trọng.

* **Giúp cho team làm việc với nhau dễ dàng hơn:** Khi làm Teamwork với nhau, mỗi người đều sẽ có một cách đặt tên class riêng và sẽ bị conflict với nhau. Việc sử dụng BEM sẽ bị loại bỏ vấn đề này vì có cấu trúc rõ ràng và dễ dàng tuân thủ khi sử dụng.
* **Modules:** Các class của mỗi block sẽ không bị ảnh hưởng gì bởi các yếu tố khác, vì thế bạn sẽ không cần quá lo CSS của class này sẽ gây ảnh hưởng đến class khác.
* **Tái sử dụng lại:** Bạn có thể soạn các block riêng biệt và sử dụng lại chúng một cách thuận tiện và giảm số lượng code CSS.
* **Cấu trúc:** BEM cung cấp cho CSS một cấu hình vững chắc, đơn giản và rất dễ hiểu

**Quy ước đặt tên của BEM**

.block {} /\* Block \*/  
.block\_\_element {} /\* Element \*/  
.block--modifier {} /\* Modifier \*/

**1. Block:** Là một thành phần của trang web hay ứng dụng đó, các thành phần của DOM cũng có thể là các block. Block ở đây thường là các thành phần header, body, content, footer. Ví dụ section td bên dưới tập hợp các block sau:

* .td\_\_container làm nhiệm vụ cân max width và padding left right  
  .td\_\_inner làm nhiệm vụ cân padding top bottom và điểm bám cho các element absolute. VD: chỉnh một button…  
  .td\_\_header chứa Sub-title.  
  .td\_\_content chứa Headline và Description.  
  .td\_\_footer chứa button CTA.

Các prefix của các class đều là td giúp chúng ta có thể nhận diện section khi muốn update mục tương ứng trong css.

**2. Elements:** Là một thành phần của một block và sẽ không tồn tại độc lập mà không có block vì được đặt bên trong nó, và chúng phụ thuộc vào parent block của nó. Trong BEM, các phần tử được biểu thị bằng dấu gạch dưới kép \_\_.

* .td\_\_headline là Text cỡ lớn. Style thường là H1 hoặc H2.  
  .td\_\_intro, .td\_\_description làm mô tả content.  
  .td\_\_image chứa ảnh.  
  .td\_\_button với style riêng cho button trong section này.

**3. Modifers:** Được dùng để thao tác thay đổi cách hiển thị trên block hoặc phần tử. Ví dụ mình muốn tạo thêm một block .block\_\_elem khác nữa và muốn làm nổi bật nó thì sẽ thêm một class .block\_\_elem--hightlight để tạo sự khác biệt đó.

**Lợi ích khi sử dụng BEM**

* Style của block không phụ thuộc vào các phần tử khác trên một trang, vì vậy bạn sẽ không bao giờ gặp vấn đề từ việc xếp tầng. Bạn cũng có thể chuyển các block từ dự án đã hoàn thành của mình sang các dự án mới.
* Có thể tạo ra các block độc lập, xây dựng thành một thư viện các block để tái sử dụng chúng. Điều này sẽ giúp cho CSS của bạn trở nên hiệu quả hơn và làm giảm lượng code CSS mà bạn sẽ phải bảo trì.
* BEM cung cấp cho code CSS của bạn một cấu trúc vững chắc mà vẫn đơn giản và dễ hiểu.

**Quy tắc, cách sử dụng BEM trong Block, Element, Modifier**

###### **ĐỐI VỚI BLOCK**

**Quy tắc đặt tên**

* Tên của block có thể bao gồm các chữ cái Latinh, chữ số và cả dấu gạch ngang
* Tạo CSS class: thêm một tiền tố ở phía trước. VD như: .block
* Bất kỳ node DOM nào cũng có thể là một block nếu như nó có một class name

VD như: <div class=”block”>…</div>

**Cách sử dụng trong CSS**

* Chỉ sử dụng bộ chọn của class
* Không dùng tên thẻ hoặc id
* Không phụ thuộc vào các block/element trên 1 trang

VD như: .block { color: #042; }

###### **ĐỐI VỚI ELEMENT**

**Quy tắc đặt tên**

* Tên Element có thể bao gồm các chữ số, chữ cái Latinh, dấu gạch ngang, gạch dưới.
* Tạo class CSS: tên block cộng với 2 dấu gạch dưới và tên Element

VD như: .block\_\_elem

* Bất kỳ một node DOM nào trong một block cũng đều có thể là một element
* Trong một block nhất định, tất cả các element cũng đều bằng nhau về mặt ngữ nghĩa

VD như: <div class=”block”>

…

<span class=”block\_\_elem”> </span>

</div>

**Cách sử dụng trong CSS**

* Chỉ sử dụng được bộ chọn class
* Không thể sử dụng tên thẻ hoặc id
* Không phụ thuộc vào các block/ element trên các trang mạng khác

VD như: Nên: .block\_\_elem { color: #042; }

Không nên: .block .block\_\_elem { color: #042; }

div.block\_\_elem { color: #042; }

###### **ĐỐI VỚI MODIFIER**

**Quy tắc đặt tên**

* Tên của Modifier có thể gồm các chữ cái Latinh, dấu gạch ngang, chữ số và dấu gạch dưới
* Tạo class CSS: tên của block hoặc element cộng hai dấu gạch ngang, cộng với tên của Modifier
* Dấu cách trong các Modifier dài sẽ được thay thế bằng dấu gạch ngang.

VD như: .block–mod { }

.block\_\_elem–mod { }

.block–color-black { }

.block–color-red { }

* Modifier là tên class mà bạn thêm vào node DOM block/ element
* Tăng thêm các Modifier class vào các nhóm block/ element mà chúng ta cần sửa đổi và giữ lại class ban đầu của block/element đó.

VD như: Nên: <div class=”block block–mod”>…</div>

<div class=”block block–size-big block–shadow-yes”>…</div>

Không nên: <div class=”block–mod”>…</div>

**Cách sử dụng trong CSS**

* Sử dụng Modifier class với bộ CSS

VD như: .block–hidden { }

* Thay đổi các element dựa trên các block có chứa modifier

VD như: .block–mod .block\_\_elem { }

* Element có Modifier

VD như: .block\_\_elem–mod { }

**Một số lưu ý khi sử dụng BEM**

*BEM* không có khả năng phản ánh cấu trúc lồng nhau của các block và các element. Giả sử: Một block có chứa element 1, element 1 chứa element 2, element 2 chứa element 3. Khi đó, tên của class của block và element nên đặt như sau:

**HTML**

*<div class=”block”>*

*<div class=”block\_\_elem1″>*

*<div class=”block\_\_elem2″>*

*<div class=”block\_\_elem3″></div>*

*</div>*

*</div>*

*</div>*

**CSS**

*.block {}*

*.block\_\_elem1 {}*

*.block\_\_elem2 {}*

*.block\_\_elem3 {}*

Đặt class như vậy, sẽ làm cho các element chỉ phục thuộc vào với block mà thôi. Sau này, nếu bạn muốn thay đổi giao diện, bạn sẽ có thể dễ dàng di chuyển vị trí của các element trong block cũng như cấu trúc của block DOM lúc này tuy đã thay đổi nhưng bạn không cần quá lo lắng về việc sửa code CSS ban đầu.

cre: techmaster.vn

[#dev\_share](https://www.facebook.com/hashtag/dev_share?__eep__=6&__cft__%5b0%5d=AZXF2K3irwmqLcwPEqMqxGgi4IR4wqmAjryipJmk7z9hnEwHCwnE8-ERvEJQDdLuuoO21uRd0xqEIGlD1hBTyBVX6u8RRGu0WPowLKFJYPnqhqPOU9-x6P3Th3didxkG62fdHeGlBEoFqO6T44QSzV84JIsMreF852WEOdz2eS1ZPMmBt0qWEpTUhKDPiL59vY4&__tn__=*NK-R)

GRID SYSTEM

Class "grid" giúp chúng ta chứa nội dung chính hoặc các thành phần chính trên layout website.

Grid được tạo dựng linh động, có thể sử dụng cho trường hợp (full-width) với class "grid" (tượng tự class "container-fluid" trong bootstrap). Và khi muốn sử dụng 1200px Grid system 12 columns các bạn có thể sử dụng 2 class là "grid wide" (tương tự như class "container" trong bootstrap).

Grid tự động thay đổi chiều rộng trên các thiết bị khác nhau. Để đơn giản hóa việc tạo responsive layout với grid system - ở đây, mình sẽ handle việc tự động thay đổi chiều rộng của "grid" với 3 breakpoints.

Row nghĩa là hàng, được sử dụng để chứa các Columns. Row giúp định hướng các Columns theo chiều ngang. Row sử dụng margin với giá trị âm sang 2 phía trái/phải để "bù trừ" cho khoảng padding trái/phải của Columns (Column padding trái/phải để tạo ra rãnh ngăn - hay còn gọi là gutters).

Column nghĩa là cột, được sử dụng để chứa nội dung/thành phần hiển thị trên website của bạn. Column sử dụng padding trái/phải để tạo nên gutters - rãnh ngăn cách giữa các column trong Grid layout. Column luôn luôn là con trực tiếp của Row.

Column được sử dụng với class "col", đi kèm theo đó là một số class "c-\*" "m-\*" và "l-\*":

**- c-\*:** nghĩa là column, prefix class này có tác dụng trên mobile. \* từ 0 tới 12. Trong đó 0 được sử dụng để ẩn column, 1 - 12 tương ứng với độ rộng chúng ta muốn sử dụng cho column (trên cơ sở 12 columns trong Grid system)  
**- m-\*:** nghĩa là medium, prefix class này có tác dụng trên Tablet.  
**- l-\*:** nghĩa là large, prefix class này có tác dụng trên PC.

# Positon

Có 4 thuộc tính top, left, right, bottom khi dùng position mới sử dụng được.

* 1. **Relative**: vị trí tương đối -> không bị phụ thuộc vào đối tượng nào khác, nó sẽ lấy chính vị trí nó đang đứng để làm gốc tọa độ
  2. **Absolute**: vị trí tuyệt đối -> bị phụ thuộc vào thẻ cha gần nhất có thuộc tính position, sẽ lấy vị trí thẻ cha để làm gốc tọa độ.

Để di chuyển đối tượng con xung quanh đối tượng cha.

* 1. **Fixed**: vị trí phụ thuộc vào khung trình duyệt

Khi 1 đối tượng phụ thuộc vào của sổ trình duyệt thì dùng fixed.

* 1. **Sticky**: vị trí bám dính (phụ thuộc vào khung trình duyệt, không khuyến cáo sử dụng) -> Có khả năng bám dính trong luồng bình thường của trang web. Khi sử dụng sticky phần tử sẽ di chuyển theo cuộn trang cho đến khi đạt 1 ngưỡng xác định, sau đó sẽ dính vào vị trí đã được chỉ định.

Tương tự như relative cho đến khi phần tử đạt đến vị trí dính sẽ chuyển sang fixed và duy trì nó trong phạm vi thẻ cha gần nhất có chứa position.

# Căn giữa

1. **Căn giữa chiều ngang**

text-aligh có tính chất thừa kế nên khi cho vào thẻ chứa hay chính nó đều có thể áp dụng.

Khi chỉ có 1 con thì nên css vào thẻ con, khi có nhiều thẻ con muốn căn giữa tất cả thì css vào cha.

1. **Căn giữa chiều dọc**

Muốn căn giữa theo chiều cao thì dùng line-height cho chiều cao bằng chính chiều cao thẻ chứa nó.

Cách khác: để căn giữa theo chiều cao là cho thẻ cha flex, thẻ con cho margin auto.

Hoặc có thể dùng trong thẻ chứa flex thuộc tính aline-item center và justtify-content

1. **Căn chính giữa màn hình**

Set cho thẻ chứa (cha) positon realative, và thẻ con là absolute để cố định sau đó sẽ xét top 50% left 50% vì top left sẽ ăn theo thẻ chứa nên sẽ dịch xuống quá 50% nên trừ đi 1 nửa chiều cao của thẻ con nên xét transform translate (-50%, -50%). Ngược lại với bottom right.

**Lưu ý:** sử dụng line-height và text-aline dùng px thì bị fix cứng quá nên độ cao sẽ sử dụng biến

# Hiển thị ảnh dự phòng khi ảnh chính lỗi

**Tìm kiếm:** placeholder image hoặc placehold.it/200x200

Thêm attribute onerror="this.src='url'" vào ngay thẻ img

Trường hợp khác sử dụng div để làm bg chứ không phải là thẻ img thì onerror không sử dụng được nó chỉ chạy được ở thẻ img hoặc các thẻ load file. Thì trong trường hợp này bg img có thể nhận được nhiều link ảnh khách nhau cách nhau = dấu , nếu link trước hỏng sẽ lấy link sau url(), url().