Kiến trúc: là một kết cấu phức tạp hay một thiết kế tỉ mỉ cho một cái gì đó.

Bạn đã bao giờ làm việc trên một dự án CSS mà dần dần nó trở thành một mớ lộn xộn, dính hết vào nhau khiến bạn mỏi con mắt, nóng cái đầu? Rất khó để xác định đoạn mã css nào ảnh hưởng tới đoạn HTML nào. Sửa một lỗi có thể làm phát sinh thêm 3 lỗi nữa hoặc nhiều hơn và có thể làm giao diện trở nên xấu xí, tệ hơn nữa là các thay đổi css nhỏ có thể làm hỏng các hàm Javascript.

Chúng tôi đã gặp phải tất cả vấn đề đó, nhưng đây là những vấn đề có thể tránh được chủ yếu bằng cách lập kế hoạch một cách cẩn thận trước khi bắt đầu một dự án. Sau đây chúng tôi sẽ nói về kiến trúc CSS.

Lợi ích của một kiến trúc css đẹp.

Lợi ích chính của việc kiến trúc css là khả năng mở rộng dự án lúc về sau. Khả năng mở rộng luôn là một thách thức đối với bất cứ dự án phát triển phần mềm nào khi mà phạm vi dự án mở rộng hoặc có thêm nhiều người tham gia vào dự án. Tính chất phân tầng và toàn bộ làm nó trở thành một công cụ mạnh nhưng cũng rất mong manh. Phân tầng(cascading) là khả năng chọn bằng các nhét thẻ của css vd: ul li, toàn bộ là khả năng ảnh hưởng tới toàn bộ thiết kế bằng các class chung. Nếu bạn đã làm việc với css một thời gian chắc chắn bạn đã từng đập đầu vào bàn khi sửa một đoạn css cho một lỗi nhưng đã làm đổ vỡ cả một vài chỗ khác.

Tóm lại một kế hoạch kiến trúc cẩn thật có thể đem lại những lợi ích sau:

- Sử dụng ít css hơn
- Ít xung đột css hơn
- Khả năng bảo trì về sau
- Tiếp cận dự án dễ hơn đối với người mới vào dự án
- Phối hợp dễ dàng hơn đối với một nhóm
- Kết thúc dự án mượt mà hơn.

Các loại quy tắc css.
Jonathan Snook đã đưa ra các khái niệm nhóm các luật css thành các chủ đề phục vụ các chứa năng khác nhau trong cuốn sách của ông: Quy hoạch và kiến trúc modules cho css (SMACSS).
Các quy tắc css được cấu trúc thành các chủ đề được xác định rõ ràng giúp tôi và đội của chúng tôi hiểu rõ hơn mục đích của từng thẻ css. Tôi sử dụng bảy loại chủ đề dựa chủ yếu vào những đề nghị trong cuốn SMACSS, chúng tạo ra những style phù hợp, gọn gàng để đưa vào một trong 7 loại này.

- Base styles (các style cơ bản)
- Objects (các đối tượng)
- Components (các thành phần)
- State (Trạng thái)
- Themes (các chủ đề)
- Utilities (Các tiện ích)
- Javascript Hooks (Các móc nối với mã javascript)

Hiểu được các loại này và mục đích sử dụng của chúng sẽ có ý nghĩa lớn tới những styles mà bạn viết.

Base styles (Các style cơ bản)
Base styles là quy tắc được tạo ra cho các phần tử cơ bản. Đây là các phong cách mặc định mà bạn muốn áp dụng trên toàn bộ trang web.

Thông thường, những thứ này bao gồm typography, box-sizing, và các phần tử mà muốn áp dụng chung trên tất cả trình duyệt. Một sai lầm phổ biến khi tạo các base styles là bạn quá nặng tay style quá sâu và tạo ra những style mặc định mà bạn không thực sự muốn. VD: Bạn có muốn bỏ bulleted ở các list không có thứ tự trên toàn bộ trang web hay bạn chỉ muốc bỏ chúng trong một số trường hợp.

Objects

Objects là các quy tắc chỉ tập trung vào cách cấu trúc và bố trí, không có một thuộc trang trí nào được sử dụng. Khái niệm này được đưa ra bởi Nicole Sullivan với mục đích tái sử dụng các mẫu cấu trúc hoặc bố trí thường được sử dụng nhiều. Tìm các mẫu cấu trúc trong thết kế của bạn và tạo ra các class objects cho chúng để có thể sử dụng trên nhiều components  hay các section của trang web. Bằng cách đặt các style cho các class object, bạn sẽ tránh được sự dư thừa, co lại kích thước file css của bạn. Các hệ thống lưới được tạo bằng tay hay mượn từng các css framework cũng thích hợp cho loại Objects này.

Components

Các components là các thành phần ui riêng biệt tách rời. Chúng là các thành phần nguyên tử để tạo lên phần lớn trang web của bạn. Một components có thể nhỏ như một button hoặc có thể lớn như cả một carousel hay slider. Chìa khóa để tạo ra các component mạnh mẽ là làm cho chúng độc lập với các component khác của trang và nó khép kín(gói gọn); có nghĩa là bạn có thể thả component lên bất cứ trang nào mà nó vẫn duy trì được cấu trúc và thiết kế (Không bị đổ vỡ);

State

Các class state là những helper class giúp sửa đổi  trạng thái của các component ví dụ như class open hay close trong các accordion hay các class is-actived trong các thẻ liên kết hoặc các thể show ,hidden để ẩn hiện các component. Thông thường những state này được điều khiển bởi javascript để thêm vào hoặc bỏ đi. Thay vì sử dụng các hàm của javascript để định kiểu bạn có thể cập nhật các lớp state này để thay đổi trạng thái của một component.

Theme

Các class theme đơn giải là để thay đổi trang trí của một component ví dụ như thay đổi các màu sắc khác nhau, font chữ khác nhau hoặc các loại trang trí khác. Các class theme có thể được sử dụng để thay đổi trang trí cho toàn bộ thành phần trong  trang hoặc chỉ một thành phần đơn lẻ. Themes có thể không được sử dụng trong mọi dự án nhưng nó vẫn tỏ ra hữu ích khi bạn cần chúng.

```
<blockquote class="c-pullquote t-light">
  <p>A great quote from someone special.</p>
</blockquote>
```

Utilities

Các class utilities là các class trợ giúp cho các mục đích đơn lẻ. Chúng có thể được sử dụng để  tinh chỉnh khoảng cách,  tăng font-size, center text, clearfix, hide ect... Các Utilities có thể giúp bạn chỉnh khoảng cách giữa các component hoặc clear float. Chúng có thể được sử dụng để thực hiện các thay đổi nhỏ ở các component hiện có mà không cần phải tạo ra một biến thể component mới.

Class Utilities

```
.u-sp {
  margin-bottom: 1em !important;
}
.u-clearfix:after {
  content: " ";
  display: block; clear: both; visibility: hidden;
  height: 0; font-size: 0;
}
.u-txt-center {
  text-align: center !important;
}
.u-txt-larger {
  font-size: 130% !important;
}
```

Sử dụng chúng:

```
<div class="promo u-sp"></div>
<div class="promo u-sp"></div>
<div class="promo"></div>
```

Javascript Hooks

Bất cứ khi nào có thể bạn hãy tách Javascript ra khỏi định kiểu thiết kế của bạn. Việc sử dụng class cho cả hai việc định kiểu và selector DOM với javascript có thể gây ra vấn đề sau này khi các css được tái cấu trúc và sự phụ thuộc và javascript không còn được rõ ràng. Thay vào đó hãy sửa dụng các class chuyên biệt chỉ để sử dụng cho mục đích select DOM với javascript

```
<button class="btn btn--buy js-buy-now">
```

Đặt tên Class
Khi đặt tên cho các class của bạn hãy đảm bảo tên class phải đủ dài để phân biệt ý nghĩa (.pullquote không viết .pq) nhưng cũng không cần chi tiết quá mức cần thiết ví dụ (.nav không viết .navigation). Khả năng đọc các tên lớp có thể tác động đáng kể trong việc giúp các thành viên trong team hiện tại và tương lai hiểu được ý đồ đàng sau các style css của bạn.
Đặt tên có ý nghĩa là một  trong những vấn đề khó khăn nhất  trong việc viết css, nhưng cũng rất hữu ích nếu bạn đặt tên một các chu đáo. Không có nhiều chủ đề chuyên sâu cho vấn đề đặt tên có ý nghĩa nhưng bạn có thể tham khảo cách đặt tên thông qua bàn viết này
https://seesparkbox.com/foundry/naming_css_stuff_is_really_hard

BEM NAMING CONVENTION

Có một quy ước tương đối hưu ích trong việc đặt tên các thành phần CSS đó là BEM. Được phát triển bởi YANDEX Công cụ tìm kiếm phổ biến của Nga. Cách đặt tên này rất đơn giản.
[BLOCK]__[ELEMENT]__[MODIFIER]
Bạn sẽ phải vật lộn với việc sử dụng những tên lớp tiết đoạn như vậy. Nhưng giá trị của BEM đem lại cho dự án sẽ vượt qua khỏi những lo ngại ban đầu của bạn. Dưới đây là một ví dụ về việc sử dụng BEM:

```
<div class="alert alert--warning">
  <h1 class="alert__title">
    <span class="alert__icon"></span>
    Alert Title
  </h1>
  <p class="alert__description">The password you entered is invalid.
</p>
</div>
```

BEM cung cấp 3 lợi ích cho dự án của bạn.

- Khả năng đọc: Sử dụng tên lớp được mô tả rõ ràng cho hầu hết các phần tử sẽ giúp người khác đọc qua tệp HTML vs CSS của bạn được dễ dàng hơn.
- Tự mô tả: Sử dụng các tên lớp phân cấp sẽ giúp các lớp của bạn trở nên rõ ràng cho việc cung cấp thông tin xem phần tử nào thuộc về  thành phần cơ sở nào.
- Tính cụ thể: Có vẻ nhưng BEM tỏ ra quá mức khi thêm một class vào mỗi phần tử trong component của bạn, nhưng bằng cách này bạn có thể giữ được độ mạnh của các bộ chọn làm cho chúng overrides được một cách trực tiếp.

Namspacing

Một best pratice khi đặt tên các class của bạn là sử dụng các tiền tố để đặt tên theo 7 loại mà chúng tôi đã nói ở trên. Các tiền tố này thêm một vài kí tự vào tên class của bạn, nhưng nó sẽ giúp bạn xác định xem class này thuộc loại nào. Đây là namespace mà tôi sử dụng:

- Object: .o-
- Components: .c-
- State: .is- or .has-
- Theme: .t-
- Utilities: .u-
- Javascript hooks: .js-

```
<footer class="c-footer">
  <div class="o-container-wide">
    <a class="c-footer__logo" href="/">The Assist</a>
    <div class="c-social c-social--follow">
      <div class="c-social__label u-txt-center">Join the
conversation</div>
      <ul class="c-social__list">
        <li class="c-social__item"></li>
        <li class="c-social__item is-active"></li>
        <li class="c-social__item"></li>
      </ul>
    </div>
    <p class="c-footer__credit"></p>
  </div>
</footer>
```

Code Style

Giống như bất kì code nào, Điều quan trọng nhất trong một dự án css là giữ phong cách code một cách nhất quán. Điều này bao gồm nhưng nguyên tắc như Khoảng trắng, thụt lề, quy ước đặt tên, comments ect... Bạn có thể tham khảo https://google.github.io/styleguide/htmlcssguide.html

Code Organization

Để tổ chức code được tối ưu, bạn nên sử dụng các bộ tiền xử lý (SASS, LESS, STYLUS) hoặc một công cụ xử lý sau để biên dịch mã của bạn postCSS. Rất nhiều lợi ích mà chúng mang lại như Biến, Function, Mixins, inport, nest. Nhưng tính năng này cung cấp một phương pháp tổ chức code tốt hơn là bạn chỉ sử dụng một mình CSS.

```scss
@import "variables";
@import "mixins";
@import "normalize";
@import "typography";
@import "headings";
@import "headings";
@import "layout";
@import "carousel";
```

Sử dụng biến khi bạn sử dụng bất kì giá trị nào nhiều hơn 1 lần. Đặt tiền tố sẽ giúp bạn xác định mục đính sử dụng của chúng

```scss
// Colors
$c-warning: Red;
$c-primary: Blue;
$c-background: White;
```

Một số biến có tính chất toàn bộ bạn nên lưu chúng tại một tệp chuyên biệt. Nhưng một số biến chỉ được sử dụng trong một thành phần nào đó thì chỉ cần định nghĩa trong tệp sử dụng chúng mà thôi. Trong scss các biến bản định có thể được định nghĩa ở ruleset lồng nhau:

```scss
.alert {
  $background-color: Red;
  $foreground-color: Cream;
  background-color: $background-color;
  color: $foreground-color;
}
```

Source Order

Vì tính chất Cascading nên thứ tự của css rất quan trọng. Nếu bạn không có chiến lược đặt thứ tự bạn sẽ luôn phải tạo ra các style để override lại các thuộc tính một cách cứng nhắc.

Gần đây, harry roberts xuất bản một phương hợp lý để thứ tự các style của bạn gọi là ITCSS, nhằm mục đích ngăn ngừa va chạm không gian. Phương pháp này rất đơn giản, bắt đầu với những style có tầm rộng nhất đặc hiệu thấp nhất và kết thúc với style hẹp và có đặc hiệu cao nhất. Điều này có nghĩa là các biến toàn cục  và cái style base bạn đặt lên đầu trong khi các utilities hay các hack ie bạn đặt ở cuối.

Harry định nghĩa 7 nhóm tệp của chúng phù hợp và sắp xếp như sau:

- Setting: Variable, setting
- Tools: Function. Mixin
- Genertic: Fontface, boxsizing, normalize, ect
- Element: Nhưng phần tử cơ bản như heading title link
- Objects: Layout và class cấu trúc
- Components: Các component riêng biệt
- Trumps: Các utilities hay các định nghĩa cuối cùng có thể override tất cả các thành phần khác

```
@import "settings.global";
@import "settings.colors";
@import "tools.functions";
@import "tools.mixins";
@import "generic.box-sizing";
@import "generic.normalize";
@import "elements.headings";
@import "elements.links";
@import "objects.wrappers";
@import "objects.grid";
@import "components.nav";
@import "components.buttons";
@import "components.promos";
@import "trumps.utilities";
@import "trumps.ie8";
```