**NOTE** for the **Build Responsive Real-World Website With HTML And CSS by Jonas Schmedtmann**

1. Color

#1098ad màu cyan đẹp

#444 màu chữ

#f7f7f7 màu nền xám nhẹ

#b7b7b7 màu xám hơi đậm

#777 lighter grey (đậm hơn nữa)

#ffe7oe nice yellow

1. Các cách select element trong CSS
2. Dùng tên element selector (hoặc list selector, descendant selector, adjacent sibling selector, pseudo element)
3. Dùng tên class selector (hoặc ID hoặc pseudo class, , attribute)
4. Dùng inheritant mechanism (hoặc universal selector)

Ngoài ra còn có thể dùng style attribute nhưng ít khi dùng.

*Thường dùng* ***class selector*** *khi chọn ít element, dùng* ***descendant selector (class + element selector)*** *cho nhiều element.*

*Thay vì dùng* ***list element****, thì ta cho element* ***2 class name*** *hoặc dùng* ***extend*** *hoặc* ***attribute*** *selector*

1. Có thể tìm nguồn gốc property nào làm cái element này có đặc tính gì đó bằng cách dùng devtool hay vì mò trong CSS.
2. Viết A element rồi B element. Khi nào B element không cần xuống dòng

Chỉ được xuống dòng thì thằng A cho phép phía sau không cần xuống dòng và đồng thời thằng B cho phép phía trước không cần xuống dòng.

Ví dụ: a element + a element. Text content + a lement.

Trường hợp phải xuống dòng: h1 + a (vì h1 bắt phải xuống dòng phía sau, dẫu cho a không cần xuống dòng phía trước).

Những thằng không bắt phải xuống dòng: inline, inline-block element, text-content, strong, em element

1. Muốn biết HTML có valid hay không thì cứ cày pretitier rồi ctrl + S, nếu valid thì nó sẽ format. Còn không thì lên trang valid HTML để tìm lỗi invalid chỗ nào. Invalid HTML có thể vẫn làm trang hiện bình thường nhưng có thể gây ra những lỗi khó chịu về sau, ví dụ không format được, không inherit đúng, tự nhiên lại hiện cái button (do không đóng button)
2. Content của 1 element có thể chỉ là mỗi text hoặc mỗi các child element khác, hoặc vừa text vừa child element khác

Content của 1 element bất kì có thể là text (thậm chí là các container element như *nav, div, footer*. Dĩ nhiên các các element như *a, p* thì có thể làm điều đó rồi.)

Content của 1 element bất kì cũng có thể là các child element khác.(như *a* element cũng có thể chứa *img* element, *li* cũng có thể chưa cả *p, img, a*. Dĩ nhiên container element như *nav* có thể làm điều đó rồi.)

Content của 1 element bất kì cũng có thể vừa text vừa các child element khác (ví dụ trong paragraph có cả text và *a* element)

* Các container element cũng có thể dùng như element bình thường (*p* element). Các element bình thường cũng có thể dùng như container, nhưng dĩ nhiên nó sẽ không làm tốt bằng container, (có giới hạn những child element nào cho phép, nếu dùng child element không cho phép thường gây ra lỗi invalid HTML dù trang web đúng như mình nghĩ.) (\*\*\*\*)

Khi content của 1 element chỉ có mỗi nội dung là text. Ta có 2 lựa chọn.

* Để đoạn text đó trong *p* element (ví dụ *footer,a*) (\*)
* Để đoạn text đó trơ trọi (ví dụ *footer,a*)

Khi content của 1 element kết hợp giữa text và các child element khác (\*\*)

* Để text đó trong *p* element. Nó sẽ nằm thành riêng 1 dòng (\*\*\*)
* Để text đó trơ trọi. Nó sẽ nằm chung 1 dòng các child element khác.

(\*) Nếu parent element là *p* thì dĩ nhiên không cần <p> <p> text </p> </p> làm chi cho nó ngộ nghĩnh.

(\*\*) Nếu parent element là p thì áp dụng quy tắc đó vẫn ra cùng kết quả nhưng không nên dùng (\*\*\*) vì nó sẽ làm invalid HTML. Đây là trường hợp đã nêu trong (\*\*\*\*)

1. Trật tự các element trong tab Style của Chrome Dev Tool

* Child element
* User Agent Stylesheet

*Inherited from:*

* Parent element
* Box model

Element is arranged according to their priority. Crossed elements is after applied elements.

1. Conflicting selectors and declarations | Priority
   1. !Important keyword
   2. Inline style
   3. IDs
   4. Classes (or pseudo classes, attribute)
   5. Element selector(or list element, descendant element, pseudo element)
   6. Universal selector
   7. Default *styles*
   8. Inherited *styles (*from body element or any other parent element)
2. Linh tinh dài dòng làm biếng đọc lại

Nếu define width của content, suy ra được width của element

Nếu define width của element, suy ra được width của content.

Khi tạo padding,margin trên và dưới thì ko làm giảm content, padding làm tăng chiều cao của element. Khi tạo padding trái và phải,margin trái và phải, thì width của content giảm, nhưng điều này chỉ đúng khi mình chưa set width cho element đó, tức là để mặc định cho nó bằng 100% browser. Haha đây chỉ là do chưa chỉnh box-sising: border box mà thôi. Nhưng mà khi không specify height và width, thì dù chế độ có box-sizing hay ko, thì khi thêm padding, element vẫn to ra, ví dụ anchor element.

(Width của element con + margin của nó) bằng (width của content mẹ).

Margin top thì đẩy chính nó xuống, margin bottom thì đẩy thằng khác xuống

Khi set width thì khoảng trống còn lại được coi là margin, nhưng khi dùng dev tool để kiểm tra thì cái margin đó ko thấy số liệu, ghi đè margin khác cũng không được.

Text align chỉ hoạt động với block element, tức là những element có width trải dài 100% parent của nó. Như link ko thể text-align:center cho đến khi chỉnh thêm display: block;

Nhưng tại sao cả cái trang của mình đều đa số là block element, width 100% browser mà ko dùng text-align, mà lại đi thu nhỏ, ko còn tính chất 100% browser của block element rồi mới căn giữa. Khác biệt giữa margin auto và text align là gì?

Khác biệt đó là text-align để center nội dung của container, trong khi margin auto để center container itself.

Ok, vậy thử text-align cả cái container của trang xem. Ủa tại sao image element và anchor element đều là inline element, tại sao nó lại text-align: center được. tại sao thằng đầu tiên trong stackoverflow nói là text-align dành cho inline element. À thì ra những thằng block level box thì nó có thể text-align center ở trong chính element đó hoặc trong cái container chứa nó, còn inline box thì chỉ có thể text-align center khi chỉnh trong cái container chứa nó thôi. Suy cho cùng tất cả, cao lắm, là chỉ có thể căn giữa nội dung trong 1 cái ô nào đó (chính element đó nếu là block, hoặc cái container to hơn) chứ ko thể căn giữa cái container.

1. Inline-block element by default

Button, img, svg

1. None vs 0 value

Nếu property đó được set mặc định, mình không biết how it works, its value thì muốn bỏ nó, hãy set its value to none;

list-style: none;

text-decoration: none;

border: none;

Còn property đó do mình set, mình biết how it works, biết các value của nó làm điều này thì set lại cho mất

margin: 0;

border-bottom-right: 0;

1. Insert img

Trong html document:

<img src= “img/phuoc.jpg” alt= “phuoc”>

<img src= “**https://**www.facebook.com” alt= “link-fb”>

Trong cascading style sheet:

.box {

background-image: url(“css-img/phuoc.jpg”);

background-image: url(“**https://**www.facebook.com**”**)**;**

**}**

\* Trường hợp chèn ảnh trong CSS có thể bỏ double quotation mark

*Tương tự khi chèn external link vào anchor element cũng để link đó có https:// (chỉ thay chữ src thành href, còn lại giống cách chèn ảnh trong html document)*

*Nhấn alt + Z để link bớt dài.*

1. How to center a div (or inline element inside a div)
2. **How to center HORIZONTALLY**

* **CSS text-align: center Property** 
  + This one is used for centering text itself, inline and inline-block element inside a block level element (center content inside a container like a div)
  + Example:

- p { text-align: center;}

- .box { text-align: center;} Then, the inline/inline-block element inside of it is centered.

* **CSS Margin Auto**
  + This one is used for centering the contaier itself (a block level element like a div)
* **Flexbox (**or **CSS Grid** if you want to center a lot of elements**)**
* **Absolute Positioning and Transform (or negative margin) (**nhưng ít dùng hơn khi horizontally)
  + Both are used for centering the both inline and block level elements

1. **How to center VERTICALLY (or BOTH VERTICALLY AND HORIZONTALLY)**

* **Flexbox (**or **CSS Grid** if you want to center a lot of elements**)**
* **Absolute Positioning and Transform (or negative margin)**

1. Những style cần chỉnh ngay cho button

Background-color

Border

Cursor (gõ cur rồi tab là được)

1. Điều thường làm đầu tiên trong CSS  
   \* { padding: 0; margin: 0; box-sizing: border-box }

body { color, font-family, line-height, font weight}

1. Quy luật phải nhớ

* Cái nào cố định thì trời sập cũng phải tuân theo chính xác
* Có khoảng trống mới align được

1. Khi nào thêm margin

* Nếu khoảng cách giữa 2 thằng là vĩnh viễn thì hay dùng margin/padding, còn khoảng cách đó là tạm thời, chẳng hạn lát nữa 1 trong 2 thằng sẽ đặt vị trí khác (dùng flex, grid, absolute position) thì không để margin/padding gì cả.
  + Một trường hợp con của việc này là không để margin/padding top/bottom của flex/grid item (và của cả first/last child của flex/grid item đó), và ko để margin/padding đối với grid/flex container
* Chỉ thêm margin vào 1 element, còn trường hợp nhiều element thì dùng gap của flex/grid
* Không thêm margin trực tiếp vào reuable element như button (anchor element)

1. So sánh padding với height/width (tương tự flex-grow với flex)

Giống nhau: đều làm cho element bự hơn

Khác:

* Padding tạo các ô có padding giống nhau, kích thước cả ô khác nhau
* Height/width tạo các ô có padding khác nhau, kích thước cả ô giống nhau

*Tương tự, flex 1 cho các flex item thì kích thước cả ô bằng nhau, còn flex 1 cho các flex item thì spacing của tụi nó bằng nhau, còn kích thước cả ô khác nhau*

1. Khi đã chỉnh box sizing border box

Khi thêm padding/border, element sẽ

* To ra nếu chưa specify dimension cho element
* Thu hẹp vùng content nếu đã specify dimension cho element

*Các grid item thường đã specify width, còn height thì chưa (vì thường dùng grid-template-columns)*

1. Default behaviour của images và text khác nhau thế nào

Khi thu hẹp viewport width

* *Text*  sẽ thu nhỏ lại, xuống dòng cho vừa viewport
* *Img* sẽ giữ nguyên, tạo thanh horizontal scroll bar

Khi thu hẹp width của cái container chứa chúng

* *Text* sẽ thu nhỏ lại, xuống dòng cho vừa container
* *Img* sẽ giữ nguyên, overflow cái container (do đó lúc này có thể dùng overflow property lên *container*)

*Vì by default, img behave khác với text ở chỗ không tự thu hẹp bản thân lại nên để có responsive web design thì phải dùng % unit cho dimension của nó, cũng như dùng max-width thay cho width.*

1. Padding hay Margin

* Khi muốn tạo spacing mà có màu nền/hình nền thì dùng padding
* Để tạo padding có 2 cách, một là thêm padding cho thằng parent, hai là thêm margin cho thằng child, và cách làm đúng là cách một

1. Cách style 1 element khác các element còn lại

* Cách 1: Tìm cách chọn riêng element đó rồi style, ví dụ đặt class name nó khác mấy thằng kia, dùng first/last child, dùng descendant selector…
* Cách 2: Cho nó có 2 cái class (trường hợp này có 2 nhánh: một là style element này khác vĩnh viễn, hay là style để kết hợp Javascript)

*Hiện nay người ta thường thay thế list selector bằng dùng 2 class name.*

1. Tác dụng của box-shadow

* Tạo bóng mờ, tạo độ sâu, standing out
* Làm product/preview/profile card (hoặc dùng background-color/ border )
* Tạo border bên trong
* Dùng cho input element
* Dùng cho trạng thái focus

1. Hover (hoặc bất cứ pseudo element khác)

* Hover thằng bố, style thằng bố
* Hover thằng bố, style bạn thằng bố/ con của bạn thằng bố
* Hover thằng bố, style thằng after pseudo element, pseudo class
* Hover thằng bố, style thằng con không được hover
* Hover thằng bố, style thằng con
  + - * hover button thì style màu icon và màu button
      * hover cái khung thì style thằng con animation di chuyển, hover cái thẻ card bố, để quay cái thẻ card side vì khi thẻ card side quay có thể ko bị hover nữa
        + *Vì khi thằng con nó di chuyển, nên hover thằng bố để đảm bảo luôn được hover*
        + *Vì khi bố có nhiều thằng con, mà muốn hover thằng bố, để mỗi thằng con chuyển động khác nhau*

1. Cách để ẩn element

* Display: none
* Opacity
  + Opacity: 0
  + Opacity: 0 + width = 0;
  + Opacity: 0 + (pointer event mouse none) + visibility = hidden
* Background-color: transparent