**CSS (3) - Định chuẩn CSS, box model và đơn vị đo**

1.1 Định chuẩn CSS, box model và đơn vị đo

**Định chuẩn CSS**

Như đã biết, nếu chúng ta **không viết mã CSS** để định dạng cho một phần tử HTML, thì trình duyệt sẽ **lấy định dạng mặc định của nó** để hiển thị trang web.

Hiện tại đang có rất nhiều trình duyệt khác nhau, và **định dạng mặc định của mỗi trình duyệt cũng khác nhau.**

Do vậy, **để kiểm soát được việc hiển thị** của từng phần tử HTML, và **đảm bảo giao diện trang web luôn ổn định trên mọi trình duyệt,** chúng ta nên định chuẩn (hay thiết đặt lại) một số định dạng CSS quan trọng. Trong tiếng Anh, quá trình này được gọi là reset CSS, hay normalize CSS.

**Một số cách định chuẩn CSS**

– Cách đầu tiên, đơn giản nhất là đặt lại giá trị của thuộc tính căn lề (margin), và dãn biên (padding) bằng 0 cho mọi phần tử (kí hiệu \*). Cách làm này không được khuyến khích vì nó ảnh hưởng đến tốc độ xử lý, trong khi còn nhiều thiết lập mặc định khác lại không được thực hiện.

    <style>

        \* {

            margin: 0;

            padding: 0;

        }

…

    </style>

– Cách 2, sử dụng các đoạn mã định chuẩn được tạo sẵn, đang có nhiều người sử dụng. Ví dụ normalize.css, reset CSS của Eric Meyer, bộ định chuẩn của Bootstraps.

– Cách 3, chúng ta tự tạo ra bộ định chuẩn.

Điều quan trọng cần nhớ **là đoạn mã định chuẩn CSS luôn được thực thi đầu tiên**, trước khi bạn thực hiện các lệnh CSS khác.

Chúng ta sẽ cùng sử dụng bộ định chuẩn normalize.css.

– Vào trang chứa mã nguồn của normalize.css. https://necolas.github.io/normalize.css/

– Bấm vào nút Download, trang web sẽ hiện thị toàn bộ mã nguồn của normalize.css > xem qua một số đoạn mã CSS để hiểu thêm > chép toàn bộ mã nguồn này, lưu vào một tập tin trong thư mục *css*, đặt tên là *normalize.css*.

– Trong tập tin HTML, khai báo để luôn thực thi đoạn mã trong *normalize.css* đầu tiên.

    <title>Cooking with Nada Surf</title>

**<link *rel*="stylesheet" *href*="css/mormalize.css">**

    <style>

Vậy là một số phần tử HTML quan trọng đã được định chuẩn. Chúng ta đã có thể viết thêm các mã định dạng CSS, mà không phải lo giao diện không theo ý muốn, hoặc hiển thị không đồng nhất trên các trình duyệt khác nhau.

**Box model**

Trong việc định dạng hiển thị, chúng ta sẽ phải **đặt các phần tử** HTML **ở một ví trí cụ thể trên giao diện, với chiều rộng, chiều dài xác định, hoặc đổ màu nền**. Để tiện lợi cho việc này, CSS đưa ra khái niệm**mô hình cái hộp (box model).**

Mô hình cái hộp là công cụ CSS sử dụng để **hiển thị các phần tử HTML**.

CSS quy ước: mỗi phần tử HTML sẽ có **một hình chữ nhật tương ứng.** Hình chữ nhật này gồm vị trí hiển thị (tọa độ x, y), chiều rộng, chiều dài, đường viền (border), căn lề (margin), dãn biên (padding), màu nền…v.v.

Một tài liệu HTML **có bao nhiêu phần tử thì có bấy nhiêu cái hộp** (trừ các phần tử trong vùng head), không phân biệt là phần tử kiểu inline hay block. Cấu trúc cha-con của tài liệu HTML cũng được thể hiện trong mô hình cái hộp, cụ thể hộp-cha (của phần tử cha) sẽ chứa hộp-con (của phần tử con).

Ví dụ, đoạn mã sau minh họa hình ảnh của box model, gồm 5 cái hộp, với cấu trúc lồng nhau.

…

   <style>

        body { border: 1px solid red; }

        h1 { border: 1px solid red; }

        div { border: 1px solid red; }

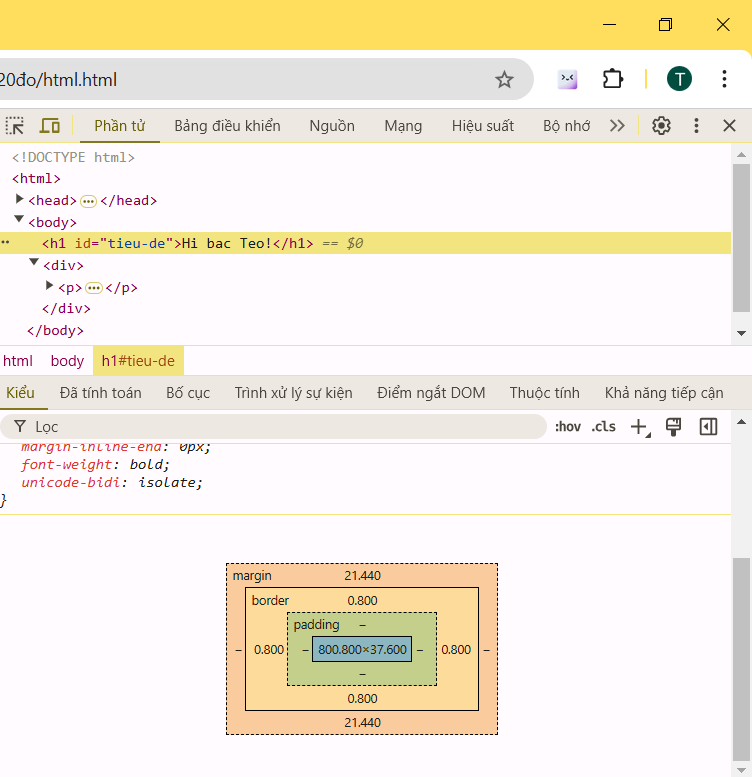
        p { border: 1px solid red; }

        em { border: 1px solid red; }

Qua quan sát, bạn sẽ thấy các hộp của phần tử kiểu block có chiều rộng chiếm hết màn hình, trong khi các hộp của phần tử kiểu inline có kích thước vừa đủ để chứa nội dung bên trong.

Sử dụng Developer tools để quan sát box model một cách trực quan.

**Đơn vị đo kích thước trong CSS**

Trong phần box model, chúng ta đã thực hiện điều chỉnh các giá trị liên quan đến khoảng cách giữa các phần tử (căn lề) (margin), độ dày của biên (border), khoảng cách giữa biên và vùng nội dung (dãn biên) (padding).

 Trong quá trình CSS cho các phần tử, chúng ta cũng phải **xác định các giá trị liên quan đến kích thước của chữ, khoảng lùi đầu hàng (indent).** Các *giá trị* luôn phải có đơn vị đo đi kèm. Vậy đơn vị đo trong CSS là gì?

CSS có nhiều đơn vị đo, chúng thuộc một trong hai loại sau**: đơn vị đo tuyệt đối (absolute), và đơn vị đo tương đối (relative).**

***Đơn vị đo tuyệt đối***

Đơn vị đo tuyệt đối là đơn vị đo tường minh, đang được dùng nhiều trong cuộc sống. Ví dụ:

– *cm*: centimet

– *mm*: minimet

– *in*: inch (1in = 96px = 2.54cm)

– *pt*: point (1pt = 1/72in)

– *pc*: pica (1pc = 12pt)

– **px: pixel, tương đương 1/96in**

Tuy nhiên**, ngoại trừ pixel**, trong làm web người ta ít sử dụng kiểu đơn vị đo này, vì chúng không linh hoạt.

***Đơn vị đo tương đối***

Đơn vị đo tương đối là đơn vị đo không tường minh, vì chúng dựa trên một độ đo khác, như dựa trên kích cỡ của phông chữ mặc định, hoặc dựa trên kích thước của phần tử cha. Ví dụ:

– *em*: tương đương với kích thước phông chữ của phần tử đang xét (hoặc phần tử cha)

– *ex*: tương đương với kích thước của chữ x

– *rem*: viết rút gọn của *root* *em*, tương đương với *em* của phần tử html (phần tử root)

– *ch*: tương đương với độ rộng của số 0 trong phông chữ hiện hành

– *vw*: tương đương với 1/100 độ rộng của cửa sổ trình duyệt (viewport)

– *vh*: tương đương với 1/100 độ cao của cửa sổ trình duyệt (viewport)

– *vmin*: tương đương với 1/100 kích thước cửa sổ trình duyệt (giữa độ rộng và độ cao, cái nào nhỏ hơn thì lấy làm chuẩn)

– *vmax*: tương đương với 1/100 kích thước của cửa sổ trình duyệt (giữa độ rộng và độ cao, cái nào lớn hơn thì lấy làm chuẩn)

– % (phần trăm): mặc dù chỉ là con số, về bản chất nó cũng có thể được xếp vào nhóm đơn vị đo tương đối.

– *fr*: cũng được xếp vào đơn vị đo tương đối, là viết tắt của fraction, nghĩa là một phần của thùng chứa (container)

***Lưu ý***

**–**Với đơn vị đo tương đối, các phần tử con không kế thừa giá trị tương đối từ phần tử cha, mà nó kế thừa giá trị đã được quy đổi (đã được tính toán).

– Đơn vị đo tuyệt đối được dùng nhiều nhất là *px*

– Đơn vị đo tương đối được dùng nhiều là: *%*, *em, rem, vw, vh, fr*

***% (phần trăm)***

Là con số, theo sau là kí hiệu %; được tính bằng cách lấy giá trị % nhân với kích thước thuộc tính của phần tử cha nó (hoặc mức cao hơn). Một số loại thuộc tính có thể sử dụng % như width, height, margin, padding và font-size.

Trong thiết kế bố cục (layout), đơn vị đo này giúp cho trang web luôn giữ được tỉ lệ cố định, sự cân đối giữa các thành phần, khi người dùng thay đổi kích thước cửa sổ trình duyệt.

Ví dụ:

     <div *style*="color: red; width: 200px">

        <p *style*="width: 50%">width: 50% == 100px</p>

        <p *style*="width: 90%">width: 90% == 180px</p>

    </div>

***em***

Là đơn vị đo tương đối. Trong kĩ thuật soạn thảo truyền thống, một *em* sẽ có kích thước đúng bằng độ rộng của chữ M (viết hoa)

Trong đặc tả của CSS, một *em* chính là độ cao của một hàng (khoảng cách giữa các baseline, không tính khoảng trắng giữa các hàng).

Như vậy, nếu văn bản có kích thước phông chữ là 16px thì một *em* sẽ là 16px, nếu phông chữ có kích thước là 12px thì một *em* sẽ là 12px. **Tóm lại, một em sẽ tương đương với kích thước phông chữ của phần tử hiện hành.**

Khi trình duyệt đã xác định được giá trị của *em*, thì có thể sử dụng đơn vị đo này cho nhiều mục đích khác, như căn lề, thụt đầu hàng, xác định kích thước cho các phần tử.

 <style>

        body { border: 1px solid red; }

        h1 {

            font-size: 32px;

            margin-left: 2em;

        }

        h2 {

            font-size: 20px;

            margin-left: 2em;

        }

    </style>

</head>

<body>

    <h1>Font-size: 32px </h1>

    <h2>font-size: 20px</h2>

</body>

Tương tự như việc sử dụng độ đo là phần trăm (%), sử dụng *em* giúp cho các thành phần duy trì được tỉ lệ cố định, cho dù kích thước của văn bản có thay đổi.

Lưu ý: 1 *em* tương đương với kích thước phông chữ hiện tại của phần tử. Vậy nếu phần tử hiện tại không thiết lập kích thước phông chữ thì sao? Khi đó 1 *em* sẽ tương đương với kích thước phông chữ của phần tử cha của nó.

***Rem***

Sử dụng đơn vị đo là *em* có khá nhiều ưu điểm, tuy nhiên trong một số trường hợp, do việc tính giá trị *em* phải dựa vào phần tử cha, điều này gây ra bất tiện cho lập trình viên khi tài liệu HTML gồm nhiều cấp

*Rem* là viết tắt của “root em”. Một *rem* sẽ tương đương với kích thước phông chữ của phần tử *html*.

nếu thiết lập kích thước cho một phần tử bất kì là *rem***, nghĩa là kích thước của phần tử đó sẽ được tính theo kích thước của phần tử htm***l* chứ không phải phần tử cha của nó.

Ví dụ,

<style>

        h1 { font-size: 3rem; }

        body {

            font-size: 10px;

        }

    </style>

</head>

<body>

    <h1>font-size: 3rem == 48px</h1>

</body>