antares;

CSS Units: px, percent, rem, em... Ano Ba Dapat Ang Gagamitin Ko?

Alin sa mga length units ng CSS ang dapat gamitin, at kailan?

Pamilyar tayong lahat sa mga percent-based na sizes, lalo na kung naabutan mo ang Responsive Web Design era. Pero hindi mo rin makakaligtaan ang pixel sizes (`px`). Pero para makuha ang tamang responsiveness, gumagamit din tayo ng iba pang mga length units. Kasama rito ang kamakailan lang ay lalong sumikat na `em` at `rem` units.

Sa artikulong ito, iisa-isahin natin ang mga length units ng CSS, kapuwa ang mga gumagana sa lahat ng browsers, at ang mga paparating pa lang sa CSS sa hinaharap (dahil hindi pa rin sapat ang mga units natin, may hinihintay pa tayong mga bago).

Una sa lahat, anong pinagkaiba ng `em` sa `rem`?

Ayon sa MDN Docs, ang `em` ay "kumakatawan sa *calculated* *`font-size`* ng isang element." Samantala, ang `rem` naman ay "kumakatawan sa *calculated* *`font-size`* ng root element, ang `<html>` sa kaso nating mga Web developer." Kaya maliwanag na halos magkapareho ang dalawang ito; nakabase sila sa `font-size`

ng isang element. In particular, sa laki ng letrang “m” sa font-size ng font na gamit mo. [citation needed]

Ano ang pagkakaiba ng `em` at `rem`? Karaniwan na, ang value ng `rem` ay naka-depende sa default font size ng browser (karaniwan nang `16px`), o sa isinet na value ng user. Samantala, ang `em` naman ay nakadepende sa `font-size` ng *mismong element*.

Bilang halimbawa, tingnan natin ito:

index.html

```
<div class="container">
</div>
```

style.css

```
.container {
  width: 3rem;
}
```

May live examples sa site.

Hindi ito madaling mapansin, pero nakasunod ang width ng `.container` sa font size ng `<html>` element. Tingnan naman natin ang `em`. Sa kasong ito, ise-set natin ang `font-size` ng `.container` para makita natin ang pagkakaiba.

index.html

```
<div class="container">
</div>
<div class="container-
lg"></div>
```

style.css

```
.container,
.container-lg {
  width: 3em;
}

.container {
  font-size: 50px;
}

.container-lg {
  font-size: 100px;
}
```

May live examples sa site.

Makikita mo ang pagkakaiba sa mga halimbawa ngayon. Parehas silang `3em` ang width, pero dahil magkaiba ang `font-size` ng dalawang ito, magkaiba rin ang naging width nila. Sinundan ng `em` ang `font-size` ng `.container`. Kung gagawan natin ito ng formula, makikita nating 3em = 3 * font-size. Ganito rin sa `rem`, ang pinagkaiba lang naka-depende ito sa `font-size` ng `<html>` element.

Puwede naman ding i-override ang size ng isang `rem`. Kailangan mo lang i-set ang font size ng `<html>` element:

styles.css

```
html {
  font-size: 20px;
}
```

Gusto kong maging responsive ang page ko. Percent lang ba ang puwede kong gamitin?

Salamat sa CSS Values and Units Module Level 4 (https://drafts.csswg.org/css-values-4/#lengths), hindi na lang tayo restricted sa percent ngayon. Nadagdagan na tayo ng apat—oo, apat—na length units: ang mga tinatawag na *viewport units*, `vw`, `vh`, `vmin`, at `vmax`.

Ang `vw` at `vh`

Ang ibig sabihin ng `vw` ay "viewport width". Para din itong percentage, pero lagi lang itong nakadepende sa width ng browser window o screen ng phone/tablet (na tinatawag ding viewport). Kung paano ka gumamit ng percent, gano'n din ang paggamit ng `vw`.

Halimbawa, kapag ginamit mo ang `50vw`, makukuha mo ang 50% ng *width ng viewport*. Kung matatandaan mo, sa `50%` makukuha mo ang kalahat ng width ng parent element, hindi ng screen.

Katulad lang din ng `vw` ang `vh`, pero tumutukoy naman ito sa "viewport height". Makukuha mo rito ang percentage ng height ng viewport.

Sa halimbawang ito, pansinin kung paanong naapektuhan ng width ng viewport ang unang box, at height naman ang nakakaapekto sa ikalawa. Pansinin din na walang pakialam ang mga box sa size ng parent element nito at lumalagpas ito sa boundaries ng box dahil nakasunod ito sa size ng viewport. Subukan mong i-resize ang browser window mo at tingnan kung ano ang mangyayari:

index.html

```
<div class="container-
vw">50vw</div>
<div class="container-
vh">50vh</div>
```

style.css

```
.container-vw {
  width: 50vw;
}

.container-vh {
  width: 50vh;
}
```

May live examples sa site.

Ang `vmin` at `vmax`

Dito nagiging interesante ang mga viewport units. Puwede mong gamitin nang halos-sabay ang `vw` at `vh`

Kapag ginamit mo ang `vmin`, pagkukumparahin ng browser ang height at width ng screen at ibibigay sa iyo ang percentage ng mas maliit na dimension. Halimbawa, kung pa-landscape ang viewport at ginamit mo ang `20vmin`, makukuha mo ang 20% ng height ng screen dahil mas maliit iyon sa width. Pero kung naka-portrait ang viewport, 20% ng width ang makukuha mo dahil mas maliit na iyon sa width.

Eksaktong kabaligtaran naman ang sa `vmax`. Ibibigay nito sa iyo ang percentage ng mas malaking dimension. Kung landscape ang orientation ng viewport, percentage ng width ang ibabalik ng `vmax`, at kung naka-portrait naman, percentage ng height ang makukuha mo.

index.html

```
<div class="container-
vmin">50vmin</div>
<div class="container-
vmax">50vmax</div>
```

style.css

```
.container-vmin {
  width: 50vmin;
}

.container-vmax {
  width: 50vmax;
}
```

May live examples sa site.

Ang iba pang mga length units

Siyempre bukod sa anim na alam na natin ngayon, may iba pang length units sa CSS.

Relative Length units

Kapag sinabing “relative”, nangangahulugan ito na hindi naka-fix ang laki nila. Nakadepende ito sa iba pang sizes sa loob ng document. Kasama rito

ang `em`, `rem`, at ang viewport units dahil hindi absolute ang values nila; nagbabago ito depende sa font size o sa viewport. Bukod sa kanila, may iba pang puwedeng magamit. In practice, wala akong masyadong nakikitang gumagamit nitong mga ‘to (posibleng may mga gumagamit nito, hindi ko lang nakikita; hindi naman ako laging nangangalkal ng source code). Pero mas magandang pamilyar tayo para alam natin kung ano’ng puwedeng gamitin kapag kinailangan.

`cap` **EXPERIMENTAL**

Kagaya ito ng `em` na nakadepende sa font at font size na gamit mo. Pero sa halip na letter “m”, nakadepende ang `cap` sa height ng capital letters.

`ch`

Nakadepende rin ito sa font size ng element. Ang `1ch` ay kasinlaki ng width ng character na `0`

`ex`

Nakadepende naman ang size nito sa height ng lowercase letters. Kinuha ito mula sa konsepto ng x-height sa typography, sa height ng letter “x” ng isang font. Karaniwan na, `1ex ≈ 0.5em`. Hindi eksaktong `0.5em` ang laki pero malapit doon.

`ic` **EXPERIMENTAL**

Nakadepende ang size nito sa width ng “水”. To be honest, hindi natin ‘to magagamit unless gagawa tayo ng website na may Chinese/Japanese /Korean version.

`lh` **EXPERIMENTAL**

Nakadepende ang size nito sa `line-height` property ng element. Kung estudyante ka at gumagawa ka ng documentation at thesis documents,

ang `line-height` ay ang height ng bawat line (weh?); sa *Microsoft Word™*, ito ang “double spacing” na tinatawag.

`rlh` **EXPERIMENTAL**

Gaya sa `rem`, ang `rlh` ay nakadepende sa `line-height` ng `<html>` element.

Absolute Length Units

Nire-represent naman ng absolute length units ang pisikal na mga sukat sa pisikal na mundo, kagaya lang din sa printed materials. Pero kahit absolute ang mga ito, nakadepende pa rin ito sa isa pang unit: ang `px`. Nakadepende ang `px` sa computation ng device, at hindi ito puwedeng kontrolin ng user o ng CSS. Dahil dito kaya tinawag itong “absolute”.

Sa mga low-dpi devices (mga screens, mapa-mobile o desktop pa man iyan), kinakatawan ng `px` ang tinatawag na *physical reference pixel*. Halimbawa, ang `1in` ay `96px`, na katumbas din ng `72pt`.

Kaya lang bilang resulta nito, hindi talaga eksakto ang mga length na gumagamit ng `in`, `cm`, at `mm`.

Sa mga high-dpi devices na gaya ng printers, eksaktong-eksakto ang length ng `px` sa diwa na `1` inch talaga ang `1in`, `1` centimeter ang `1cm`, at `1` milimeter ang `1mm`. Sa mga kasong ito, `1px` = `1/96` inch.

NOTE: May mga users na nag-a-adjust ng font size ng browser para mas madaling magbasa. Dahil hindi ito sinusunod ng absolute length units, magkakaproblema ka sa accessibility kung ginagamit mo ang mga ito para sa mga size ng text. Kaya naman mas maganda kung relative length units ang gagamitin mo.

`cm`

One centimeter. `1cm = 96px ÷ 2.54`

`mm`

One millimeter. `1mm` = 1/10th ng `1cm`

`Q` **EXPERIMENTAL**

One quarter ng millimeter. `1Q` = 1/40th ng `1cm` or 1/4th ng `1mm`

`in`

One inch. `1in` = `2.54cm` = `96px`

`pt`

One point. `1pt` = 1/72nd ng `1in`

`pc`

One pica. `1pc` = `12pt` = 1/6th ng `1in`

Browser compatibility

Napansin mo siguro na may mga units tayong “experimental”. Pag-usapan muna natin ang Web standards. Ginagawa ito ng W3C at WHATWG. At simula nang lumabas ang “CSS3” kung tawagin, nahati na ang CSS sa iba’t ibang modules na may sari-sariling version (na dahilan din kung bakit hindi na magkakaroon ng CSS4, dahil hiwa-hiwalay nang nagpo-progress ang mga modules ng CSS). Ang mga length units na ginagamit natin ay nasa ilalim ng *CSS Values and Units Module*, na ang latest version ay Level 3. Habang sinusulat ko ito, may ginagawa nang Level 4.

Ang mga units na “experimental” ay either nasa Level 4, o nasa Level 3 na pero hindi pa ini-implement ng lahat ng browsers. Kaya ang mga “experimental” ay hindi siguradong gagana sa lahat ng browsers, lalo na ang mga galing sa Level 4. Sa ngayon, nasa *editor’s draft* status pa ang Level 4 ng Values and Units module kaya kahit kailan puwede itong mabago.

Kasama sa mga units na na-define sa Level 3 na sigurado nang ii-implement ng browsers sa hinaharap ay ang `ch`, `rem`, viewport units, at ang `Q` unit, although

hindi pa lahat ng browsers ay nai-implement ang `Q` unit, gaya ng Edge, Edge Mobile, Safari, Safari iOS, at Samsung Internet.

Ang `lh` at `rlh` naman ay kasama sa Level 4. At habang isinusulat ko ito, wala pang browser na nag-i-implement ng mga ito, maliban sa `rlh` na mayro'n na sa Edge browser at Edge Mobile.

Sa kahit anong feature ng Web platform na hindi mo sure kung naka-implement na sa mga browser, makakatulong sa iyo ang Can I Use? (https://caniuse.com) I-search mo lang sa site na ito ang feature ng Web na gusto mong makita at ibibigay nito sa iyo ang browsers na nag-implement na noon, at kung ilang percent ng users ang makakakita ng feature na iyon kapag ginamit mo na. (Hindi ito sponsored ng Can I Use, talagang useful lang siya sa akin kaya nire-recommend ko)

Alin sa mga ito ang dapat kong gamitin?

Kung habol mo ang responsiveness, laging effective pa rin ang percentage units. Pero hindi ito laging praktikal. Puwede mo ring gamitin ang viewport units dahil supported na ito ng karamihan sa mga browser. Ingat lang sa `vmin` at `vmax` dahil hindi pa ito supported ng Edge browser (pero malapit na rin dahil sa paglipat ng Edge sa Chromium backend). Pero mas maganda kung pag-aaralan mo ang Intrinsic Web Design tools na gaya ng CSS Grid at Flexbox dahil ginawa talaga ang mga ito para sa responsive web pages.

Sa accessibility naman, mas preferable na gumamit ng relative units kaysa sa absolute units. Sa susunod, puwede bang gamitin mo ang `rem` at `em` sa pagse-set mo ng `font-size` bilang respeto sa mga gagamit ng size mo? Tandaan na hindi lahat ng users sa Web ay walang problema; may mga users na differently abled gaya ng mga bulag at malabo ang paningin, at may mga users din na hindi kumpleto ang device (e.g. walang mouse, kaya keyboard lang ang gamit). Kaya bilang pagpapakita na may pakialam din tayo sa kanila, mas mabuting maglaan ng space para sa accessibility, at nagsisimula ito sa paggamit ng relative length units.

Pero may mga times pa rin na useful ang absolute length units. Halimbawa, masyadong malaki ang `rem` at `em` para sa `border` at `border-radius` sa mga card UI, kaya mas mainam gamitin ang `px`. At kung publisher ka naman at ginagamit mo ang CSS para gumawa ng aklat (oo, puwedeng gamitin ang CSS para gumawa ng mga libro; tingnan ang CSS Paged Media sa kahit na anong blog), makakatulong nang malaki ang `in`, `cm`, `mm`, `pc`, at `pt` (at alam niyo ba na `pt` ang gamit na unit sa font size ng mga word processor gaya ng MS Word at WPS?).

At the end of the day, nakadepende pa rin sa iyo kung alin ang mga gagamitin mo. Matututuhan mo kung kailan gagamitin ang mga ito kung susubukan mo silang gamitin sa susunod mong experiment sa mga Web page. Kaya laging mag-practice sa paggawa ng mga Web page.

Tingnan ang pinakabagong version ng artikulong ito sa
https://celestialcinnamon.github.io/antares-blog/tl/CSS-Units-px-percent-rem-em-Ano-Ba-Dapat-Ang-Gagamitin-Ko/

See me outside

- Antares on Facebook (https://facebook.com/antaresprogramming)
- Antares on Github (https://github.com/celestialcinnamon/antares-blog)

- See me on Facebook (https://facebook.com/dorkas.rubio)
- See me on Twitter (please don't)
- See me on Instagram (https://instagram.com/melancholicapoptosis)
- See my portfolio (https://celestialcinnamon.github.io)
- Send me an email: francoisoibur21@gmail.com

Ang Antares Programming

Ang Antares Programming ay isang blog para sa mga Pilipino tungkol sa mga bagay tungkol sa Web at software development na hindi madalas maituro sa mga university at college. Dahil kinikilala ng Antares Programming ang epekto ng wikang kinalakhan o *mother tongue* sa pagkatuto, karamihan ng mga artikulo sa site na ito ay nasa Filipino. Umaasa ang writer nito na darating ang panahon na magkakaroon ng mas maraming materyal sa iba pang mga wika ng Filipinas. Pero sa ngayon, sapat na ang pagsisikap na ito.

Nga pala, hindi laging ganito "kalalim" (kapormal) ang Filipino sa site na ito. ☺