Colors i Mides

Colors

Els **colors** són molt utilitzats en CSS, ja sigui per al color del text, el color de fons, els degradats, les ombres, les vores... Hi ha diverses maneres de definir els colors en CSS, cadascuna amb els seus pros i contres.

La propietat `color` defineix el color del **text**. És bastant senzill. El que és més important són els diferents tipus d'**unitats de color** disponibles.

Noms de colors

CSS proporciona 145 noms de colors, des dels més bàsics (negre, blanc, taronja, groc, blau...) fins a els més específics (gespa, orquídia, carmesí...).... obviament **in english**:

```
body{ color: black;}
a{ color: orange;}
```

Com que els noms dels colors són difícils de recordar i perquè probablement voleu colors molt específics, els noms de colors no s'utilitzen sovint.

Colors en format RGB

Els monitors d'ordinador, televisors, telèfons mòbils utilitzen el model de color RGB per mostrar colors. Bàsicament, cada color es defineix per una combinació de vermell, verd i blau. Hi ha 256 valors possibles per a vermell, verd o blau. Com que els ordinadors comencen a comptar a 0 (zero), el valor màxim és 255.

Tenint en compte que un color és el resultat d'una *combinació* de vermell, verd i blau, i com que cadascun d'aquests 3 colors té 256 valors possibles, hi ha `256 * 256 * 256 = 16.777.216' colors possibles disponibles.

Com que el model RGB està directament relacionat amb com es representen *físicament* els colors, s'ha convertit en una unitat de color CSS.

Exemples de RGB

Per exemple, el color groc de la M de McDonald's te 255 quantitats de vermell, 199 de verd i 44 de blau:

```
a{ color: rgb(255, 199, 44);}
```

El color negre no és cap quantitat de vermell, verd o blau:

```
body{ color: rgb(0, 0, 0);}
```

A l'altre costat de l'espectre, el blanc és la quantitat completa de cada vermell, verd i blau:

```
body{ color: rgb(255, 255, 255);}
```

Colors en format RGBA

La unitat de color `rgba` és `rgb` a la qual afegim un valor **alfa** (que va de 0 a 1, en valors decimals), que defineix com de transparent és el color:

```
/* Un color negre lleugerament transparent. */
body{ color: rgba(0, 0, 0, 0,8);}
```

El propòsit que un color sigui transparent és barrejar-se amb el fons i, per tant, semblar lleugerament diferent segons el context. És especialment útil per als **colors de fons**.

Colors en hexadecimal

Els colors en CSS també es poden definir amb **valors hexadecimals**, com `#db4e44`.

Per entendre què són els valors hexadecimals, mirem com funcionen els binaris i els decimals:

Considereu els números del 0 al 9 i les lletres A-F com a **símbols**.

Els humans utilitzen el sistema **decimal**. Tenim 10 símbols per formar nombres.

En **hexadecimal**, tenim 16 símbols per formar nombres. Com que 0-9 no són prou símbols, també fem servir A-F. I comença a zero.

Unitats de mida

Hi ha moltes propietats CSS que requereixen **unitats de mida**:

- `font-size` defineix la mida del text
- `border-width` defineix el pes de les vores dels elements
- `margi` defineix l'espaiat entre elements
- `left/right/top/bottom` permet posicionar i moure elements

Les unitats més utilitzades són:

- `px` per a píxels
- `%` per percentatge
- `em` per a la mida relativa al valor `font-size` del pare.

Píxels

Com que les pantalles d'ordinador utilitzen píxels per mostrar el contingut, és la **unitat de mida més comuna en CSS**.

Es pot utilitzar per fixar l'**amplada** d'un element:

```
body{ width: 400px;}
```

O defineix la **mida del text**:

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

Els píxels en CSS són senzills perquè defineixen **valors absoluts**: no es veuen afectats per altres propietats CSS heretades.

També s'utilitzen àmpliament amb finalitats de **posicionament** i **espaiat**.

Percentatges (I)

Els percentatges són **unitats relatives**: es basen en el pare i/o l'avantpassat de l'element.

Per exemple, els elements a nivell de bloc com els paràgrafs ocupen naturalment l'**amplada total disponible**. La següent regla CSS els canviarà la mida a **la meitat** de l'amplada disponible.

```
p{ width: 50%;}
```

Percentatges (II)

Els percentatges poden ajudar a establir altres propietats CSS, com ara la mida del text:

```
strong{ font-size: 200%;}
```

En aquest cas la mida de la font serà 2 vegades la mida que tenia assignada inicialment.

EM (I)

`em` és una unitat **relativa**: depèn del valor de la `font-size' de l'element.

Per exemple, si el pare té una mida de font de `20px` i apliqueu `font-size: 0.8em` a un element secundari, aquest element secundari representarà una mida de font de `16px`.

No confongueu la unitat de mida CSS `em` i el selector CSS `em`, que s'orienta als elements HTML `<em>`

EM (II)

La unitat `em` és interessant ja que definiu les mides de lletra dels elements HTML *relatives* entre si. Per dissenyar una pàgina web agradable i fàcil de llegir, necessiteu una profunditat visual constant. Per exemple, voleu que el vostre `<h1>` sigui el doble de gran que el vostre text corporal, el vostre `<h2>` només 1,5 vegades més gran i el vostre peu de pàgina una mica més petit. Això es podria aconseguir fàcilment en CSS:

```
body{ mida de la font: 16px;}
h1{ font-size: 2em;} /* = 32px */
h2{ mida del tipus de lletra: 1,5 em;} /* = 24 píxels */
footer{ font-size: 0.75em;} /* = 12px */
```

EM (III)

Si decidiu canviar la mida del text del cos, les mides relatives dels encapçalaments i de la barra lateral **canviaran en conseqüència** i la vostra pàgina web es mantindrà **visualment equilibrada**.

Amb només canviar un valor, tots els altres valors s'alteren:

```
cos{ mida de la font: 20px;}
h1{ font-size: 2em;} /* = 40px */
h2{ mida de la font: 1,5 em;} /* = 30 píxels */
footer{ font-size: 0.75em;} /* = 16px */
```

REM

La unitat `rem` és semblant a `em`, però en comptes de dependre del valor del *parent*, es basa en el valor de l'**element arrel**, que és l'element `<html>`.

```
html{ mida de la font: 18px;}
body{ font-size: 1rem;} /* = 18px */
h1{ font-size: 2rem;} /* = 36px */
h2{ mida del tipus de lletra: 1.5rem;} /* = 27px */
```

REM & EM (I)

La diferència entre `rem` i `em` és que els valors `rem` són **fixos** mentre que els valors `em` es poden *multiplicar* entre si.

Si configureu el vostre `html{ font-size: 18px;}`:

- `2rem` sempre serà igual a `36px`, sense importar on utilitzeu al vostre CSS
- `2em` sempre serà igual a **doble** la `mida de la lletra` dels pares, de manera que no necessàriament `36px`

REM & EM (II)

Exemple ràpid on "2em" és diferent de "2rem":

```
html{ mida de la font: 20px;}
p{ mida de la font: 0.8rem;} /* = 16px */
p span{ font-size: 2em;} /* = 16px * 2 = 32px */
p fort{ font-size: 2rem;} /* = 20px * 2 = 40px */
```

El `span` es basa en el valor de la mida del tipus de lletra `p` mentre que el `strong` es basa en el valor de la mida del tipus de lletra `html`.

Altres tipus de mides

Hi ha moltes més unitats de mides disponibles, com per exemple pulçades (`inches`) o mides *relatives* com `vw` i `vh` , que fan refereència a la mida de la finestra on se està visualitzant la pàgina:

https://www.w3schools.com/cssref/css_units.asp

Quina unitat utilitzar?

Recomanaria **píxels** per començar: com que són valors absoluts, no es veuen afectats pel context de l'element. Són senzills, permeten establir la mida del text, dimensions de la imatge, amplada de la vora, coordenades de posició...

Els valors **Percentatge** i **em** es poden utilitzar al costat dels píxels, especialment per a mides de text relatives.