UD02 - Conceptes bàsics

Índex

CSS - Conceptes Bàsics

CSS - Cascade Style Sheets

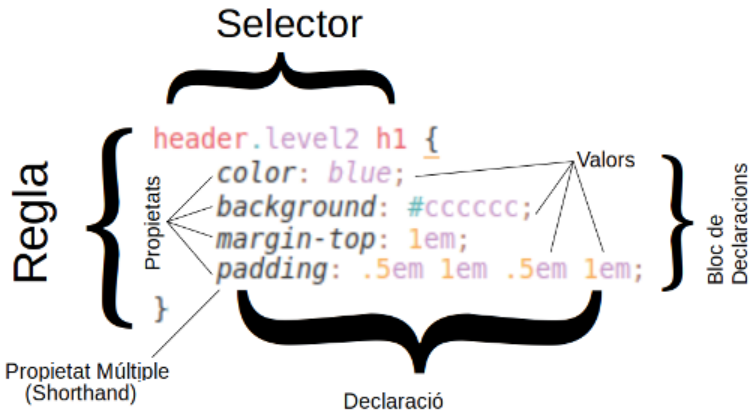
Nivells d'estandardització:

1. Editor's Draft (ED)
2. Working Draft (WD)
3. Transition – Last Call Working Draft (LCWD)
4. Candidate Recommendation (CR)
5. Transition – Proposed Recommendations (PR)
6. Recommendation (REC)

Font: https://css-tricks.com/css-standards-process/

Nomenclatura

- **Propietat:** Designa un paràmetre o característica que desitjam ajustar a l'element o elements seleccionats.
- **Valor:** El valor que li volem assignar a una *propietat*.
- **Declaració:** És el conjunt d'una *propietat* i el seu *valor*, separades per dos punts (":").
- **Bloc de Declaracions:** És una llista d'una o més *declaracions*, separades per punt i coma (";"), i envoltes en claus ("{ ... }").



- **Selector:** És una expressió que ens permet seleccionar un o varis elements del DOM als que volem aplicar un *bloc de declaracions*.
- **Regla:** Es el conjunt format per un *selector* i un *bloc de declaracions*.
  - També pot tenir varis selectors separats per comes (",").
- **"Multipropietats" (shorthand property):**
  - Ex: `font` = `font-style` + `font-weight` + `font-size` + `line-height` + `font-family`.
  - Ex: `margin` = `margin-top` + `margin-right` + `margin-bottom` + `margin-left`.
  - Acostumen guardar coherència:
  - Ex: Top - Right - Bottom - Left ("TRouBLe" / Des d'amunt i en sentit de les agulles del rellotge).
  - Valors per defecte "simètrics"
  - En cas de dubte, consultar documentació.
  - **Efecte col·lateral:** SEMPRE estableixen totes les propietats (encara que s'ometin).

Regles

Valors Especials:

- *Initial*: És el valor per defecte de l'atribut a un determinat tipus d'element.
- *Auto*: Significa que el navegador pot ajustar-lo automàticament en funció de certes premises preestablertes.

Selectors

Selectors Bàsics

- `tagname`
- `.class`
- `#id`
- `\*`

Combinadors (Combinators)

Nom	Sintaxi		Exemple
Compound (composició)	(res)	`(a)(b)`	`h1.page-header`
Descendant (descendent)	(espai)	`(a) (b)`	`header h2`
Child (descendent directe)	(>)	`(a)>(b)`	`li>ul`
Adjacent Sibling **(posterior)**	(+)	`(a)+(b)`	`th+tr`
Following Sibling (el següent)	(~)	`(a)~(b)`	`tr.today~tr`

Selectors de Pseudo-Classe (Pseudo-class Selectors)

Els selectors de pseudo-classe ens permeten adreçar elements per alguna característica extrínseca a ells com ara la ubicació en relació a altres elements o determinats estats transitoris com ara si ténen el focus del teclat o si el cursor del ratolí es troba a sobre d'ells.

La llista completa de selectors de pseudo-classe es pot trobar al MDN sota:

- https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes.

Selectors de Pseudo-classe Genèrics

`:first-child`	`:first-of-type`
`:last-child`	`:last-of-type`
`:only-child`	`:only-of-type`
`:nth-child(an+b)`	`:nth-of-type(an+b)`
`:nth-last-child(an+b)`	`:nth-last-of-type(an+b)`
`:not(<selector>)`	`:root`
`:focus`	`:hover`
`:empty`	`":blank"` (proposta)

Notes:

- `:root` sel·ecciona el tag `html` en documents HTML.
- `:empty` no pot contenir ni un espaï en blanc o salt de línia.
- `":blank"` de moment només és una proposta: no suportat per cap navegador a dia d'avui.

Selectors de Pseudo-classe Específics per a Formularis

Alguns selectors de pseudo-classe es refereixen específicament a característiques de camps de formularis (tags `<input>`, `<textarea>`, `<button>`, `<select>` i/o `<option>`, segons el cas):

`:enabled`	`:disabled`
`:valid`	`:invalid`
`:required`	`:optional`
`:checked`	

Avís: Alguns d'ells han estat introduïts o redefinits recentment pel que no funcionaran com s'espera en alguns navegadors.

Haurem de consultar cadascun a http://caniuse.com per a saber exactament com estan de suportats als distints navegadors.

Selectors de Pseudo-element

Els pseudo-elements, a diferència de les pseudo-classes, no adrecen elements (tags) realment existents al document, sinó que *virtualment* en creen de nous en un lloc determinat o sobre una part determinada de l'element seleccionat.

Per exemple, `p::first-letter` ens permetria aplicar un estil distint **únicament** a la primera lletra de cada paràgraf com si aquesta estigués continguda en el seu propi *subtag*.

Es distingeixen de les pseudo-classes per anar prefixats per dos caràcters de "dos punts" (`::`) en comptes d'un sol com els pseudo-elements.

Avís: Ens podem trobar però casos en que s'hagin especificat amb només ':' ja que així fou com s'especificaren originalment i per ara tots els navegadors els reconeixen també. Nosaltres **sempre** els escriurem com a `::after` i no `:after`, per exemple. I si detectam codi antic o algun cas en que, per error, hem fet servir la versió incorrecta, la rectificarem.

::before i ::after

Creen un pseudo-element que passa a ser el primer o darrer fill, respectivament, de l'element seleccionat.

- Es poden utilitzar per pre/su-fixar texts o imatges.
- **És obligatori** definir la propietat `content`, encara que sigui amb un espaï en blanc, ja que **del contrari no apareixeran**.

::first-letter i ::first-line

Permeten adreçar, respectivament, la primera lletra o la primera línia de text, d'un element.

::selection

Permet adreçar text (i sub-elements) que estiguin sel·leccionats per l'usuari.

Només permet unes poques propietats. Incloses `color`, `background-color`, `cursor`, and `text-decoration`.

Selectors d'Atribut

Els selectors d'atribut permeten seleccionar elements en funció dels seus atributs HTML i ténen la mateixa especificitat que els selectors de classe (0,1,0).

- `[attr]`: Existència de l'atribut (l'elements tenen l'atribut especificat, independentment del seu valor).
  - Exemple: `input[disabled]`
- `[attr="value"]`: Concordancia de valor.
  - Exemple: `input[type="button"]`
- `[attr^="value"]`: Concordancia inicial (que el valor "comença per...")
  - Exemple: `a[href^="https"]`
- `[attr$="value"]`: Concordancia final (que el valor "acaba en...")
  - Exemple: `a[href$=".pdf"]`
- `[attr*="value"]`: Concordancia parcial (l'atribut conté el text especificat).
  - Exemple: `*[class*="sprite-"]`.
- `[attr~="value"]`: Concordancia d'element en llista (d'elements separats per espaï)
  - Exemple: `div[class="menu"]` (en aquest cas equivaldria a `div.menu`, però no esta limitat a l'atribut "class").
- `[attr|="value"]` Igualtat o concordància de prefix (prefix = fins al primer "-").
  - Exemple: `div[lang|="es"]` (concordaria amb `lang="es"`, `lang="es-ca"`, etc..)

Selectors d'Atribut *Case-insensitive*

Els selectors d'atribut *tradicionalment* distingeixen entre majúscules i minúscules, pel que, per exemple, `a[href^="mailto://"]` **no seleccionaria, per exemple `<A HREF="MAILTO://someone@example.com>Email me</A>`**.

Recentment s'ha especificat un modificador que ens permet convertir els selectors anteriors en *Case Insensitive* només afegint una "i" al final.

Per exemple: `a[href^="mailto://" i]` sí seleccionaria l'enllaç anterior.

AVÍS: Encara hi ha molts de navegadors que no uporten el modificador *i*. Per aquest motiu és millor procurar no dependre'n mentres ens sigui possible (en el cas de l'exemple anterior, procurant normalitzar tots els enllaços a minúscules (o majúscules).

Cascada (WIP)

1. Origen del Full d'Estil
2. Especifitat del Selector
3. Ordre al Codi Font

Origen

L'Origen fa referència a la procedència de les regles d'estil.

Principalment tenim dos origens:

1. Els *Author Styles* que son els que definim nosaltres als fulls d'estil.
2. Els *User Agent Styles* que son els estils que el navegador du predefinits de sèrie i gràcies als quals, fins i tot sense cap full d'estil (d'autor), un document HTML es renderitza amb uns estils mínims:
   - Marges
   - Lletres més grosses als títols (h1, h2, h3...)
   - El color blau i el subratllat dels enllaços (a).
   - Etcètera...

Addicionalment, alguns navegadors permeten configurar un full d'estil personalitzat per a l'usuari. En tal cas tindriem un tercer origen (els *User Styles* o Estils d'Usuari) que **es situarien al mig dels altres dos**.

Les declaracions especificades *inline* al propi *tag* HTML (mitjançant l'atribut *class*) s'apliquen directament a l'element i ténen preferència sobre qualsevol altra independentment del seu origen.

Exemple: `<h1 style="color:red;">`

Els estils del *User Agent* son els que ténen menys preferència de tots. Això ens permet canviar-los a gust.

...però també ens permet **no fer-ho** si els estils que proveeix el navegador per defecte son suficients per a nosaltres.

Però els *User Agent Styles* poden variar d'un navegador a l'altre. El que **en alguns casos pot arribar a ser un problema:**

Una diferència mínima en el tamany (o tipus) de lletra, per exemple, pot fer que un text passi d'ocupar-nos una línia a dues.

Per evitar aquest problema, es sol incorporar un fill de ja predefinit amb els estils que volem **per a tots** els navegadors.

Si no volem fer-ho manualment, en podem fer servir un de ja fet, com normalyze.css.

!important

El modificador `!important` al final d'una *declaració* CSS ens permet botar-nos la preferència d'origen fent que la regla s'apliqui fins i tot si hi ha una altra regla amb més preferència que la contradiu.

El problema però és que si dues declaracions en conflicte ténen la clausula `!important` aleshores prevaldrà de nou la preferència d'origen (i la resta que veurem després) que puguin aplicar.

Una forma senzilla d'entendre-ho és pensar que en realitat tenim 6 possibles origens en comptes de 3:

1. **Author !important**
2. **(*User Styles*) !important**
3. **User Agent !important**
4. Author
5. (*User Styles*)
6. User Agent

Especificitat (WIP)

Donades dues declaracions conflictives i del mateix origen s'aplica sempre la més específica.

> **EXEMPLE:** Imaginem un extens manual de circulació on apareixen, no necessàriament en aquest ordre i pot ser múltiples vegades en distints capítols, les següents regles:
>
> - Les bicicletes circularan per l'esquerra.
> - Tots els vehicles circularan per la dreta.
>
> ...Si una bicicleta és un vehicle, per ón ha de circular una bicicleta?

Com veieu intuitivament el concepte és molt senzill: Els *selectors* de l'exemple serien "Les bicicletes" i "Tots els vehicles". I tots hem entès què ha de fer cadascun encara que les regles siguin contradictòries.

Quan es tracta de *Selectors CSS* però, la cosa es complica. Considerem per exemple el següent fragment:

```
<style>
    h1 {
      font-size: 16px;
    }
    .big {
      font-size: 18px
    }
</style>
<h1 class="big">Títol</h1>
```

...quin tamany de lletra s'aplicarà al text "Títol"?

Per poder decidir-ho necessitam unes regles precises que es puguin aplicar a qualsevol selector (pensau que un selector pot ser la composició de molts altres).

I per poder definir aquestes regles classificarem els selectors segons tres criteris:

- Nombre d'Identificadors.
- Nombre de Classes.
- Nombre de Tags.

Per representar aquesta classificació es sol fer servir la notació `(nI,nC,nT)` ón *nI* seria el nombre d'identificadors, *nC* el nombre de classes i *nT* el nombre de tags.

 Les *Pseudo-Classes* i els *Pseudo-Elements* **ténen la mateixa especificitat que les *Classes*.** Pel que els comptarem també com a tals.

A la taula següent podem veure uns quants exemples de selectors i la classificació que els correspondria.

Selector	Nº d'IDs	Nº de Classes	Nº de Tags	Notació
html body header h1	0	0	4	0,0,4
section header.section-header h1	0	1	3	0,1,3
.page-header .title	0	2	0	0,2,0
#sidebar	1	0	0	1,0,0

Ordre

Finalment, quan l'Origen i l'Especificitat coincideixen, la declaració que apareixi **després** guanya.

Així:

- Si el nostre document té varis fulls d'estil (ja siguin externs o inline), les regles que apareixin als fulls d'estil inclosos més avall en el document, tendran preferència sobre les que apareixin als anteriors.

- Si dues regles en conflicte apareixen al mateix fitxer (o dins el mateix tag `<style>...</style>`), la que apareixi més al final tindrà preferència.

 No és extrany tampoc especificar la mateixa propietat, fins i tot dins la mateixa regla. **Exemple:**

```
.menu { display: block; display: flex; }
```

...en aquest cas guanya **sempre** la segona declaració. Però en un navegador que no suporti *Flexbox*, aquesta seria ignorada, pel que la primera ens serveix per definir un valor per defecte quan el navegador no suporta quelcom que volem fer servir (*fallback*).

Herència (WIP)

Variables

Una variable CSS es defineix mitjançant una declaració a la qual, el nom de la propietat comença per dos guions ("--").

Exemple:

```
--home_background_color: cyan;
```

Funcionen com una declaració normal i son heretades pels elements descendents amb la única diferència que no tenen cap efecte per si mateixes.

Per llegir-les fem servir la funció `var()`.

Exemple:

```
background-color: var(--home_background_color);
```

Aritmètica CSS3

Unitats

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

Longituds absolutes

Unitat	Descripció
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

Els 'px' son en realitat una unitat relativa i no es corresponen amb els pixels físics del dispositiu sino que van en funció de la definició d'aquest amb la finalitat que les mesures en 'px' siguin el més semblants possibles en el món real independentment del dispositiu de visualització.

Longituds relatives

Unitat	Descripció
em	Relative to the font-size of the element (2em means 2 times the size of the current font)
ex	Relative to the x-height of the current font (rarely used)
ch	Relative to width of the "0" (zero)
rem	Relative to font-size of the root element
vw	Relative to 1% of the width of the viewport*
vh	Relative to 1% of the height of the viewport*
vmin	Relative to 1% of viewport's* smaller dimension
vmax	Relative to 1% of viewport's* larger dimension
%	Relative to the parent element

Càlculs (WIP)

(funció calc())

Regles "AT" (@)

Referències

- Especificitat: https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity
- Media print:
  - Designing For Print With CSS: https://www.smashingmagazine.com/2015/01/designing-for-print-with-css/
  - Maquetació avançada d'impressió
  - (Desgraciadament cap navegador suporta massa més que establir els marges).

(TO-DO !!) * PostCSS: https://postcss.org/. - Autoprefixer: https://github.com/postcss/autoprefixer.

Javascript - jQuery

Referències

- You might not need jQuery