CSS

lloc: CIFP Francesc de Borja Moll
Curs: Llenguatges de marques i sistemes de gestió d'informació
Llibre: CSS
Imprès per: Joan Llompart Socias
Data: divendres, 5 novembre 2021, 10:33

Taula de continguts

1. Fulls d'estil: CSS

Per tal d'evitar que l'HTML fos el responsable de la part estètica i visual de la web, es van idear els que s'anomenen, fulls d'estil, és a dir, es va dissenyar el llenguatge CSS (*Cascading Style Sheets*).

- **Com ja hem comentat, al codi HTML col·loquem el contingut, és a dir, què ha de visualitzar-se,**
- **Mentre que amb CSS definim la presentació i l'estil, és a dir, com ha de visualitzar-se.**

Recomanació: L'adequat quan treballem amb CSS, és escriure el codi en fitxers independents, que tindran extensió ***.css****. No convé col·locar codi CSS per tant dins d'arxius HTML, és a dir, s'ha d'evitar col·locar estils en etiquetes <style> en el propi codi HTML.*

Introducció

Les regles CSS, per poder definir el disseny de la pàgina, utilitzen:

- **Selectors**: Mitjançant ells, podem especificar a quins elements de la pàgina ens estem referint. Són les referències a les etiquetes (per exemple, id's i/o classes d'HTML.
- **Declaracions:** Són les unitat bàsiques de CSS, el que significa que no es pot emprar res més petit. Consisteix bàsicament en l'assignació d'un valor a una propietat. Es podria dir, de forma col·loquial que és la resposta a una pregunta: De quin color hauria de ser els fons? *per exemple: color:red*

Com és una regla CSS?

En primer lloc, si volem mitjançant CSS modificar l'estil d'una o diverses etiquetes HTML en una pàgina web, haurem d'indicar a quina etiqueta o etiquetes afecta la modificació.Aquesta part del codi són els selectors. Per a indicar-los, excepte casos especials, s'escriurà el nom de l'etiqueta o una referència a ella. per exemple, si en aquesta part escrivim h1, l'estil que apliquem afectarà a totes les etiquetes <h1> que hi hagi en la pàgina.

Després, s'haurà d'escriure què és el que volem canviar d'aquesta etiqueta i com. Tot això formarà un bloc, que anomenarem declaració.
En CSS s'empren els signes "{" i "}" per a delimitar aquest bloc; Dins de les claus escriurem què és el que volem canviar i el nou valor que li donarem. Al que volem canviar li diem propietat, i ho indiquem mitjançant una paraula clau. Cada propietat té un valor (o varis). El valor indica la variació de la propietat, és a dir, quan o en quina ha canviat la propietat.

Exemple:
Si volem canviar la grandària dels títols amb etiqueta h1: h1 { font-*size : 12px } En aquest exemple h1 indica les etiquetes a les quals hem d'aplicar aquest estil; font-*size indica la propietat (la grandària de lletra), mentre que 12px és el valor, és a dir la variació de la propietat (el canvi de grandària). Com pot observar-se la propietat i e valor estan separats pel signe de dos punts.

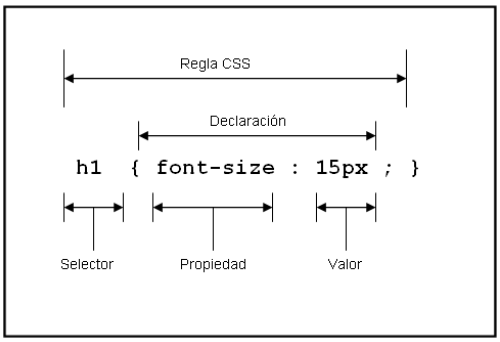
Dins d'una mateixa declaració (signes de claus), pot haver-hi més d'una propietat amb el seu valor, aquestes han d'anar separades pel signe de punt i coma.

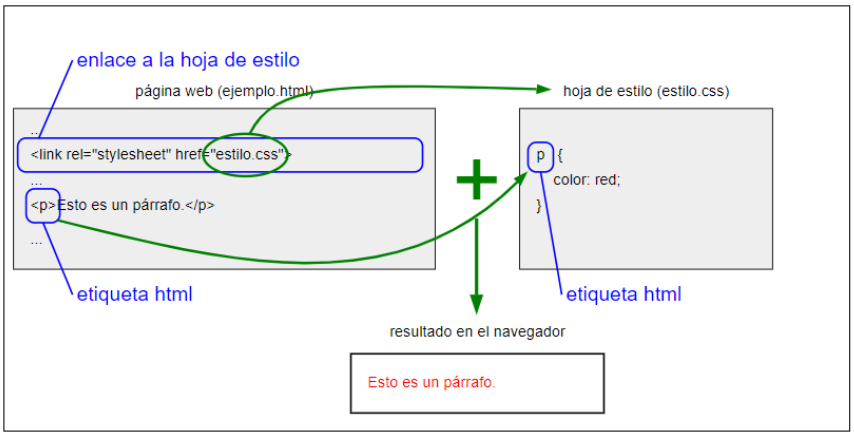
Exemple:
h1 { font-*size : 12px; color : blue; } En aquest exemple les etiquetes h1 reben dues propietats, amb la primera canviem la grandària de lletra, i amb la segona canviem el color de lletra. El signe de punt i coma, es pot posar al final de l'última propietat, encara que no és obligatori resulta convenient, per si volem afegir alguna propietat més.

El llenguatge CSS es basa en una d'estructura, que podem resumir de la següent forma: **selector { propietat : valor; propietat : valor; ... }**

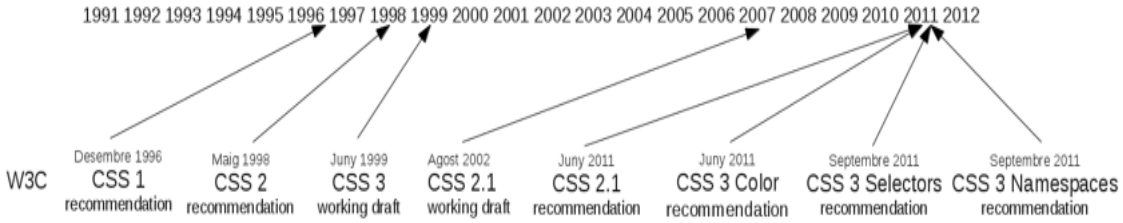
Conceptes importants:

- **Regles:** Cada regla o conjunt de regles consisteix en un o més selectors i un bloc de declaració (o blocs d'estil).
- **Dins dels blocs** existeixen les propietats o elements, els quals tenen un valor determinat.
- **Els estils** s'apliquen als elements del document que compleixin amb el selector que els precedeix.
- **Cada bloc** d'estils es defineix entre claus, i està format per una o diverses declaracions d'estil amb la sintaxi.





- **Evolució històrica del CSS**
  - La primera versió del CSS o Cascading Style Sheets, va aparèixer en 1996.
  - Després el maig de l'any 1998 es va publicar l'estàndard CSS versió 2.
  - El 2008, se'n va fer una revisió i es va publicar el CSS versió 2, revisió 1 i es va conèixer com CSS2.1.
  - Actualment està en vigor la versió 3, el CSS3. En aquest cas l'especificació està dividida en mòduls, alguns dels quals ja han aparegut estàndards i en d'altres encara s'hi està treballant, és a dir, per una part tenim estàndards i per una altra una tecnologia en continu desenvolupament.



Evolució de les versions de CSS

Com podeu observar a partir de la versió CSS3, els estàndards es tracten en mòduls per separat, com els **selectors** o **namespaces.**

- **Avantatges de l'ús de fulls d'estil**

  Emprar fulls d'estil ens proporciona els avantatges següents:

  - Possibilitat de mantenir el codi.
  - En el camp de disseny, el CSS és més potent que les marques de disseny que ofereix L'HTML.
  - El CSS és un llenguatge senzill.
  - Es poden especificar diferents fulls d'estil per a un sol document HTML. Per exemple, podem tenir l'estil per a la pàgina web quan es visita amb el navegador i l'estil per a quan volem imprimir aquesta pàgina.
  - Els fulls d'estil es poden reutilitzar des de diferents documents HTML.

- **Inconvenient de l'ús de fulls d'estil**

El gran inconvenient dels fulls d'estil és que no tots els navegadors es comporten de la mateixa manera davant del mateix full d'estil. Això es deu al fet que alguns navegadors no compleixen els estàndards establerts i, amb això, obliguen el programador a codificar diferents fulls d'estil (un per a cada navegador). Tot i això, en els últims anys els navegadors cada vegada més, s'acosten més a complir els estàndards proposats.

1.1. La sintaxi

Recordatori:

Un full d'estil és un conjunt de regles que defineixen l'estètica final dels documents HTML. Com ja hem dit abans aquestes regles estan formades per un se de declaracions.

- **Un selector**: Ens serveix per definir a quin o a quins elements volem aplicar les declaracions de la regla.
- **Una declaració**: Està formada per una propietat amb el seu valor associat. Les declaracions són les diverses característiques que han de complir els elements qu selector. A cada propietat de les declaracions, s'hi posa un valor.

- ### La sintaxi genèrica

```
selector{
     declaració_1
         ...
     declaració_n
    }

on: la sintaxi de cada declaració_i és: propietat_i:valor_i;
```

Exemple: si volem que tots els paràgrafs tenguin lletra de mida 10pt i un fons de color gris, definirem:

```
p {
  font-size: 10pt;
  background-color: gray;
   }
```

En aquesta regla, p és el selector i té dues declaracions: font-size:0pt; i background-color:gray.

En la primera declaració font-size: 10pt, font-size és la propietat i 10pt és el valor.

En la segona declaració background-color:gray, background-color és la propietat i gray és el valor.

- ### Les regles arrova

Hi ha un conjunt de regles especials que s'anomenen regles arrova (*at-rules* en anglès o regles-at). Aquestes regles es caracteritzen perquè comencen pel caràcter @. Algunes d'aquestes regles (@import o @namespace) han d'aparèixer al principi de la fulla d'estil, i les altres (@font-face, @media, etc.), poden posar-se a qualsevol part del full d'estil.

Vegem quines són aquestes regles i com s'utilitzen:

- **@import:** La regla *@import* ens serveix per incloure, en el nostre full d'estil, fulls d'estil externs. Això és útil en determinats casos, per exemple, quan es treballa amb una web gran i complexa, l'arxiu CSS és torna molt gran i mal de manejar; en aquest cas seria interessant emprar @import.

  Exemple:

  Si, per exemple, volem incloure en el nostre full d'estil totes les propietats que hi ha en un document anomenat "nousestils.css", hem d'escriure la línia següent:

  ```
  @import "nousestils.css";
  ```

- **@media:** La regla *@media* serveix per diferenciar per quin mitjà s'ofereixen les propietats que conté aquesta regla. Són les Media Queries (que veurem més endavant de forma més detallada) i són una de les grans avantatges de CSS3, ja que permeten saber quin sistema està visualitzant la pàgina web, i en funció d'això s'aplicaran unes regles d'estil o unes altres. És un dels millors recursos de que disposen els dissenyadors per fer els seus llocs responsives.

  La sintaxi genèrica és la següent:

```
@media mitjà {
   propietats
}

on mitjà pot ser print (per imprimir) o screen (per mostrar per pantalla), entre d'altres.
```

Exemple: Volem que, quan imprimim el document HTML, la lletra tengui una mida de 12pt, però que, quan es vegi per pantalla, tengui una mida de 14pt. També volem que en els dos casos, l'alçada de la línia sigui d'1.4. En el nostre full d'estil hem d'introduir el codi següent:

```
@media print {
  body{ font-size: 12pt }
}

@media screen {
  body { font-size: 14pt }
}

@media screen, print {
  body { line-height: 1.4 }
}
```

Avui en dia, amb l'augment dels tipus de dispositius, fa que tenguem moltes mides de pantalles diferents (smartsphones, tablets, ordinadors, etc.), i aquesta regla és molt útil per aplicar diferents regles CSS segons el tipus de pantalla o la seva orientació. Aquest tipus de programació se l'ha anomenat Disseny Adaptatiu (en anglès *Responsive Web Design* o RWD), i a la combinació de la regla @media amb les condicions que podem afegir, se l'ha anomenat *media-queries*.

- **@font-face:** Especifica una font no inclosa en el navegador. a regla CSS. ës a dir, permet definir una tipografia i importar-la pel seu ús a una pàgina web. Abans d'existir aquesta regla @, només es podia definir una llista de families en ordre descendent de prioritat amb la regla "font-family".

Exemple: En aquest exemple, en la regla @font-face estem definit una font. Per tal de definir-la li posem un nom. Aquest nom es posa amb la propietat font-family i en aquest cas hem triat el nom DeliciousRoman. A més, és imprescindible especificar on es troba aquesta font, per tal que el navegador la pugui descarregar i emprar (això es fa amb la propietat src)

```
@font-face {
  font-family: DeliciousRoman;
  src: url("Delicious-Roman.otf");
            }

  p {
       font-family: DeliciousRoman, Helvetica, Arial, sans-serif
    }
```

Si hem definida la font, després la podrem emprar en qualsevol regla.

La seqüència de font dins l'atribut font-family, vol dir s'intentarà primer aplicar la primera font, si no es pot aplicar, se intentarà amb la segona, i així successivament.

- **@charset:** Especifica quin és el joc de caràcters que farem servir dins del fitxer CSS:
  - @charset "UTF-8": Activa el joc de caràcters pel full d'estil a Unicode UTF-8.
  - @charset 'ISO-8859-15': Activa el joc de caràcters pel full d'estil Latin-9 (Llengües de l'oest d'Europa, amb el símbol de l'euro).

- **@supports:** Ens permet detectar si el navegador de l'usuari suporta o no les noves funcionalitats del CSS. El seu funcionament seria:

Exemple: El CSS aplica les regles només si el navegador suporta display: flex. En cas contrari ho ignora per complet.

```
/* si el navegador suporta display: flex */
@supports (display: flex) {
/* Llavors aplica les regles: */
    .element {
      display: flex;
      ... Més regles
    }
 }
```

- **@page:** La fem servir per modificar propietats CSS a l'hora d'imprimir un document, però hem de tenir en compte que només pot actuar sobre un nombre de propietats: sobre els marges, les línies vídues, línies orfes i els salts de pàgina.

Exemple:

```
@page {
  margin-left: 3cm;
}

@page :left {
   margin-left: 4cm;
}
```

Llista de propietats CSS estables

Pots consultar per exemple les següent pàgines web, on trobaràs totes les propietats de CSS que pots fer sevir:

- *https://www.w3schools.com/cssref/*
- https://www.mclibre.org/consultar/htmlcss/css/css-propiedades.html

Els comentaris

Com en tot llenguatge, la llegibilitat és imprescindible si treballen en grup. És per això que CSS ofereix una manera de comentar els fulls d'estil. Si en un full d'estil volem posar comentaris destinats a l'aclariment del codi CSS, hem de fer servir la sintaxi següent:

```
/* comentaris */
```

Exemple:

```
/* Estil per a totes les capçaleres del document */

h1 {
  text-align: center; /* Text centrat per destacar la importància de la capçalera*/
  color: red;         /* Color de lletra vermella per emfatitzar el text */
}
```

Ubicació del CSS

Al mateix document HTML

Afegint les propietats CSS directament a l'element emprant l'atribut style.

Exemple: Volem que un paràgraf estigui centrat amb lletra vermella. Hem de tenir en compte que aquesta forma d'incloure codificació CSS s'ha de fer servir puntualment, o per fer proves, però no permet reutilitzar el codi en diverses etiquetes.

El codi html seria el següent:

```
<p style="text-align:center; color:red">Paràgraf centrat vermell</p>
```

Emprant CSS:
A la capçalera del document HTML i podem posar les diferents propietats CSS dins de l'element <style> que està ubicat dins de l'element <head>. Per exemple, si volem que tots els paràgrafs estiguin centrats i amb lletra vermella, el codi és el següent:

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8" lang="ca">
    <title>Document HTML </title>
    <style>
        p {
          text-align:center;  color:red
          }
    </style>
  </head>
  <body>
    <p>Paràgraf centrat vermell</p>
    <p>Paràgraf centrat vermell</p>
    <p>Paràgraf centrat vermell</p>
  </body>
</html>
```

- ***En un document extern***

Posam totes les propietats dins d'un document amb extensió .css i des del document HTML enllacem aquest full d'estil amb l'ajuda de l'etiqueta <link>, dins la capçalera <head>.

Exemple: Si volem tenir tots els paràgrafs centrats amb lletra vermella, el document HTML és:

El codi htmlñl seria:

```
<!doctype html>
  <head>
    <meta charset="UTF-8" lang="ca">
    <link rel="stylesheet" href="estils.css" />
  </head>
  <body>
    <p>Paràgraf centrat vermell</p>
  </body>
</html>
```

El fitxer estils.css té el contingut següent:

```
p {
   text-align:center;  color:red;
   }
```

1.2. Atributs globals: class i id

Els atributs **"id"** i **"class"** serveixen com a identificadors a HTML, són molt importants perquè els elements HTML puguin ser apuntats i personalitzats amb estils mitjançant CSS.

- **Atribut id**
  - L'atribut global id, defineix un identificador únic, és a dir, ha de ser únic en tot el document. El seu propòsit és identificar l'element quan es vincula.
  - El valor d'aquest atribut no ha de contenir espais en blanc (espais, tabulacions, etc.).

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>identificadors</title>
    <style>
        div#principal {
          border: 1px solid blue;
          padding: 20px;
          margin: 0, 25%;
          font-family: Georgia, 'Times New Roman', Times, serif;
        }
    </style>
  </head>
  <body>
    <div id="principal">
        <p>El aroma de los libros es obra de una pluma envolvente y divertida, es una pequeña joya de delicadeza y de hum
or, pero sobre todo es un homenaje al poder de la lectura. Con un ritmo de comedia, encuentra sus mejores momentos en la
dinámica entre los personajes.</p>
    </div>
  </body>
</html>
```

- **Atribut class**
  - Si es vol donar o definir un estil distint a elements amb la mateixa etiqueta, es pot fer emprant l'atribut "class". Aquest atribut es pot assignar a qualsevol element d'una pàgina web.
  - Encara que l'especificació no posa requisits en el nom de les classes, s'anima als desenvolupadors web a utilitzar noms que descriguin el propòsit semàntic de l'element, en lloc de la presentació de l'element. Els noms semàntics segueixen sent lògics encara que la presentació de la pàgina canviï.

Codi

Fragment HTML:

```
<p>- Haga el favor de poner atención en la primera cláusula porque es muy importante. Dice que ... la parte contratante de la primera parte será considerada como la parte contratante de la primera parte. ¿Qué tal, está muy bien, eh?</p>

<p class="aviso">- No, eso no está bien. Quisiera volver a oírlo.</p>
```

CSS

```
p.aviso {
  color: red;
}
```

Resultat

- Haga el favor de poner atención en la primera cláusula porque es muy importante. Dice que ... la parte contratante de la primera parte será considerada como la parte contratante de la primera parte. ¿Qué tal, está muy bien, eh?
- No, eso no está bien. Quisiera volver a oírlo.

Exemple:

```
<html>
  <head>
    <meta charset="utf-8" />
    <title>classes</title>
    <style>
        .blava {
            background-color: blue;
        }
    </style>
  </head>
  <body>
    <button class="blava">Botó en fons blau</button>
  </body>
</html>
```

A diferència de "id", "class", es pot utilitzar més d'una vegada a la pàgina.

Exemple:

```
<html>
  <head>
      <meta charset="utf-8">
      <title>Atributs global: id i class</title>
  </head>
  <body>
      <h1 id="titol-principal" class="verd-turquesa">Pàgina sobre bicicletes</h1>
      <h2 id="subtittol-principal" class="tittol">Anem amb bici per la ciutat</h2>
      <p class="descripcio">Al mercal trobam molts de tipus de bicicicletes.</p>
      <p class="descripcio">Bicicletes per atletes, per nins, per somniadors, etc.</p>
      <p class="descripcio">La bicicleta ens permet der lliures</p>
  </body>
</html>
```

Exemple:

```
<!DOCTYPE>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Aplicació amb id i class</title>
    <style>
      div#principal {
          border: 1px solid blue;
          padding: 20px;
          margin: 0 25%;
          font-family: Tahoma, sans-serif;
      }
      .destacat {
          font-size: 1.3em;
          color:red;
      }
    </style>
  <body>
    <div id="principal">
      <p>Aquesta és la secció principal. Tot el contigut apareixerà dins un requadre.</p>
      <p>Seleccionam un <div> amb un atribut id., i un <span> amb un atribut class. Els estils són:</p>
      <ul>
          <li>Propietat <span class="destacat">border</span></li>
          <li>Propietat <span class="destacat">padding</span></li>
          <li>Propietat <span class="destacat">margin</span></li>
          <li>Propietat<span class="destacat">font-family</span></li>
      </ul>
    </div>
  </body>
</html>
```

1.3. Selectors / Pseudoclasses/ Pseudoelements

Al fulls d'estil (CSS), els selectors són la part de les regles que indiquen al navegador a quins elements s'ha d'aplicar les propietats incloses dins de les declaracions.

- ## Selectors bàsics
  - ### Selector universal
    - S'utilitza per a seleccionar tots els elements de la pàgina.
    - Se indica mitjançant un asterisc (*).
    - No s'empra habitualment, ja que és difícil que un mateix estil es pugui aplicar a tots els elements d'una pàgina.

Exemple: eliminar el marge i l'emplenament de tots els HTML:

```
* {
  margin: 0;
  padding: 0;
}
```

  - ### Selectors de tipus o etiqueta
    - Selecciona tots els elements de la pàgina on l'etiqueta indicada, coincideix amb el valor del selector.
    - Per utilitzar aquest selector, només és necessari indicar el nom d'una etiqueta HTML (sense els caràcters < i >) corresponent als elements que es volen seleccionar.

Exemple: aplica diferents estils als titulars i als paràgrafs d'una pàgina.

```
h1 {
  color: red;
}

h2 {
  color: blue;
}

p {
  color: black;
}
```

    - Si es volen aplicar els mateixos estils a dues etiquetes diferents, es poden encadenar els selectors, separats per comes:

Exemple: mostra com els encapçalaments <h1>, <h2> i <h3> comparteixen els mateixos estils.

```
h1 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}

h2 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}

h3 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}


En aquest cas, CSS permet agrupar totes les regles individuals en una sola regla amb un selector
múltiple. Es posen tots els selectors separats per comes:

h1, h2, h3 {
  color: #8A8E27;
  font-weight: normal;
  font-family: Arial, Helvetica, sans-serif;
}
```

Selectors de classe

- S'utilitzen quan volem que alguns elements amb la mateixa etiqueta a la qual s'ha donat ja un estil, en tengui un altre de forma excepcional.

Exemple:

Si tenim el següent estil:

```
h1, h2, h3 {
   color: #000033; <!-- assigna el color blau a h1,h2,h3-->
}
```

Però volem que alguns dels encapçalaments <h2> es vegin en vermell i negreta. En aquest moment és quan podem emprar els selectors de classe. El procediment seria el següent:

1.- En primer lloc, assignarien (dins del codi HTML), a <h2>, l'atribut de classe propi:

```
class= '...':  <h2 class="vermell-negreta">Encapçalament vermell i negreta</h2>
```

2. En segon lloc, a la fulla d'estil, escrivim la instrucció especial que s'ha d'aplicar a la classe que acabam de definir, és a dir, afegim el nom de la classe al selector de l'element separat per un punt i sense espais en blanc:

```
h2.vermell-negreta {
   color: #330000;
   font-weight: bold;
}
```

Podem anar més enllà, podem generalitzar més l'estil per si a aquest, el volem aplicar a altres etiquetes. Es tracta de no vincular la classe a cap etiqueta:

```
.vermell-negreta {
   color: #330000;
   font-weight: bold;
}
```

I d'aquesta manera, es podrà assignar aquesta classe a qualsevol etiqueta HTML:

```
<p class="vermell-negreta">Aquest paràgraf està en vermell i en negreta</p>
```

Selectors Id

- A vegades, serà necessari aplicar estils a un únic element de la pàgina. Encara que es podria emprar un selector de classe per aplicar estils a un únic element, els selectors id són més eficients en aquest cas.
- Aquests selectors permeten seleccionar un element de la pàgina a través del valor del seu atribut.
- Aquests tipus de selectors només seleccionen un element de la pàgina perquè el valor de l'atribut id no es pot repetir dins dos elements diferents de la una mateix pàgina.
- La sintaxi és molt similar a la dels selectors de classe, amb la diferència que utilitzen el símbol # en lloc del punt(.).

Exemple:

```
<h1 id="verd-negreta">Títol en verd i en negreta</h1>
```

La instrucció en el CSS seria:

```
h1#verd-negreta {
  color: #003300;
  font-weight: bold;
}
```

Selector descendent

- Selecciona els elements que es troben dins d'altres elements.

Exemple: El selector del següent exemple, selecciona tots els elements de la pàgina que es troben dins d'un element amb etiqueta <p>.

```
p span {
color: red;
}
```

Si el codi HTML és:

```
<p>
 ...
 <span>text1</span>
 ...
 <a href="">...<span>text2</span></a>
 ...
</p>
```

On: *El selector: p span, selecciona tant "text1" com "text2". El motiu: és que en el selector descendent, un element no ha de ser descendent directe de l'altre, l'única condició és que un element ha d'estar dins d'un altre element (sense importat el nivell de profunditat en el que es trobi). A la resta d'elements de la pàgina que no es trobin dins de l'element <p>, no se'ls hi aplicaria.*

- Els selectors descendents sempre estan formats per dos o més selectors separats entre sí per espais en blanc, on el darrer selector indica l'element on s'apliquen els estils i tots els selectors anteriors indiquen el lloc on s'ha de trobar aquest element.

Exemple: 4 selectors

```
p a span em {
text-decoration: underline;
 }
```

Els estils s'apliquen als elements de tipus que es troben dins d'elements de tipus , que a la vegada es troben dins elements de tipus <a> els que es troben dins d'elements de tipus <p>.

Combinació de selectors

- Es permet la combinació d'un o més tipus de selectors per restringir l'àmbit d'aplicació de les regles del CSS.

Exemple:

Si tenim un estil que només selecciona aquells elements amb una classe anomenada especial (class="especial") que es troba dins qualsevol element amb un classe anomenada avis (class="avis"), tendriem:

```
.avis .especial {
    ...
}
```

Si es modifica per:

```
div.avis span.especial {
    ...
 }
```

Aquest estil, defineix que el selector només selecciona aquells elements de tipus amb un atribut class="especial" que estiguin dins qualsevol element de tipus que tengui un atribut class="avis".

Selectors d'atributs

- Els elements HTML també es poden seleccionar pels seus valors d'atribut o per la presència d'un atribut.
- L'atribut s'especifica entre claudàtors i el selector d'atributs pot adoptar diverses formes:
  - **[nom_atribut]**: Selecciona els elements que tenen l'atribut nom_atribut", independentment del seu valor.
  - **[nom_atribut = "valor"]**: Selecciona tots els elements que tenen l'atribut donat, el valor del qual és el valor de la cadena.
  - **[nom_atribut ~ = "valor"]**: Selecciona tots els elements que tenen l'atribut donat, el valor del qual conté el valor de la cadena separat per espais en blanc.
  - **[nom_atribut | = "valor"]**: Selecciona els elements l'atribut "nom_atribut" tingui exactament el valor "valor" o comenci per "valor" seguit d'un guió.
  - **[nom_atribut ^ = "valor"]**: Selecciona els elements l'atribut "nom_atribut" tingui un valor amb prefix "valor".

Exemples:

```
/* Fa referència als elements "p" que tenen un atribut "exemple" */
p[exemple] {
  color: red;
}
----------------------------
/* Fa referència als elements "a" amb un atribut "href" que coincideixi amb "https://exemple.es" */
a[href="https://exemple.es"] {
  color: green;
}
----------------------------
/* Fa referència als elements "a" amb un atribut "href" que comenci per "im" */
a[href^="im"] {
   color: grey;
}
----------------------------
/* Fa referència als elements "a" amb un atribut "href" que acabi per ".es" */
a[href$=".es"] {
 font-style: italic;
}
----------------------------
/* Fa referència als elements <p> que tenguin un atribut "class" que contengui la paraula "exemple" */
p[class~="exemple"] {
 padding: 2px;
}

-----------------------------------------------
Exemple d'aplicació:

CSS:

li[class="valor1"] {
   background-color: yellow; <!--Elements "li" amb "class" igual a valor1-->
}

li[class~="valor1"] {
   color: red; <!-- Elements "li" que "class" tengui la paraula valor1-->
}


HTML:

<h1> Selectors d'atributs </h1>
<ul>
  <li>Primer element</li>
  <li class="valor1">Segon element</li>
  <li class="valor1 valor2">Tercer element</li>
  <li class="valor1valor2">Quart element</li>
</ul>
```

- ## Selectors combinadors
  - ### Combinador de germans adjacents
    - El combinador "+", selecciona germans adjacents, és a dir, que el segon element segueix directament al primer i tots dos tenen el mateix element pare.
    - Ha d'aparèixer al DOM (arbre d'elements) immediatament després de tancar el primer element germà i no hi pot haver-hi cap altre germà que les separi.

```
Sintaxi: Element-germà + Element-germà directe o contigu.
```

Exemple:

CSS:

```
h2 + p {
   font-size: 14px
} /* Aplica a tots els elements <p> que són germans directes d'un <h2>*/


---------------------------
```

HTML:

```
<html>
   <head>
      ...<!--cridam al css-->
   </head>
   <body>
      <h2> Fills adjacents </h2>
      <p> Estic fent una prova </p> <!-- Els css aplicarà damunt aquest element <p>-->
      <h2> Pareix que queda bé </h2>
      <address> Contacte< /address>
      <p> agrupació....</p> <!-- a aquest <p> no s'aplica ja que no és germà contigu de <h2>-->
      ....
   </body>
</html>
```

Combinador general de germans

- El combinador "~", selecciona germans, és a dir, que el segon element segueix al primer (no necessàriament de forma immediata), i els dos tenen el mateix pare.

Sintaxi: `Element-germà ~ Element-germà`

Exemple:

CSS:

```
p ~ span {
   background-color: orange
}
/* Aplica a tots els elements <span> que segueixen a un <p>*/


-------------------------
```

HTML:

```
<html>
   <head>
      ...<!--cridam al css-->
   </head>
   <body>
      <h1> Germans </h1>
      <p> Estic fent una prova </p>
      <h3> Pareix que queda bé </h3>
      <span> Si, definitivament, queda bé </span> <!-- Els css aplicarà damunt aquest element <span>-->
      ....
   </body>
</html>
```

Combinador de fills

- El combinador ">", selecciona els elements que són fills directes del primer element.

Sintaxi: `Element-pare > Element-fill-directe`

Exemple:

```
ul > li {
   color : yellow;
}
/* La regla aplicarà a tots els elements <li> que són fills directes d'un element <ul>*/
```

- ## pseudoclasses

  - Les pseudo-classes permeten seleccionar elements:
    - segons un estat específic d'un element.
    - en funció de la posició de l'element en el document.
  - Tendeixen a actuar com si haguessis aplicat una classe en una part determinada del document i, sovint, ajuden a reduir l'excés de classes.
  - Proporcionen un marcat més flexible i fàcil de mantenir.

Sintaxi: Les pseudoclasses són paraules clau que comencen amb dos punts.

```
:nom de la pseudo-classe
```

Exemples (segons l'estat):

:active - Coincideix amb un element que s'està activant actualment. S'empra als botons i als enllaços. Normalment vol dir, que el botó del ratolí s'ha premut però encara no s'ha deixat anar.

```
a:active {
  color: red;
}
```

```
<html>
...
<body>
...
   <a href="https://www.google.es"> Google </a>
</body>
</html>
```

:checked - Coincideix amb un botó d'opció, una casella de selecció o una opció dins d'un element de selecció que està marcat o seleccionat.

```
:checked {
  margin-left: 25px;
  border: 1px solid blue;
}
```

:focus - Coincideix amb un element que té el focus actualment. Normalment s'utilitza per a botons, enllaços i camps de text.

```
input:focus {
  color: red;
}
```

:hover - Coincideix amb un element sobre el qual es troba el cursor del ratolí. Normalment s'utilitza per a botons i enllaços, però es pot aplicar a qualsevol tipus d'element

```
a:hover {
  color: orange;
}
```

:valid, :invalid - S'utilitza amb elements de formulari mitjançant la validació HTML5. La pseudo-classe :valid, coincideix amb un element que actualment és vàlid segons les regles de validació i, :invalid no coincideix amb un element que actualment no és vàlid.

```
input:valid {
  background-color: powderblue;
}
```

:visited - Coincideix amb un enllaç l'URL del qual ja ha estat visitat per l'usuari. Per protegir la privadesa de l'usuari, el navegador limita l'estil que es pot fer en un element coincident amb aquesta pseudoclasse.

```
a:visited {
  color: green;
}
```

Exemple (segons la posició en el document): Volem que els primers paràgrafs dels article, es vegin amb un tipus de font arial i en cursiva (per exemple), podem crear una classe.

CSS

```
.primerparagraf {
   font-family: arial;
   font-style: italic;
}
```

HTML:

```
<article>
   <p class="primerparagraf">Primer paràgraf del primer element article que volem canviar.</p>
   <p>Paràgraf que no volem canviar.</p>
</article>
<article>
   <p class="primerparagraf">Primer paràgraf dels segon element article que volem canviar.</p>
   <p>Paràgraf que no volem canviar.</p>
</article>
```

==========

Hi pot haver un problema, per exemple, si després volguéssim afegir una paragraf a la part inicial d'un article, hauriem de moure la classe (per tant canviar el codi HTML). Per evitar-ho podem emprar un selector de pseudoclasse: per exemple: "first-child", que sempre seleccionarà el primer fill.

CSS:

```
article p:first-child {
 font-family: arial;
 font-style : italic;
}
```

HTML:

```
<article>
   <p> Paràgraf afegit i que ara volem canviar </p>
   <p class="primerparagraf">Primer paràgraf del primer element article que volem canviar.</p>
   <p>Paràgraf que no volem canviar.</p>
</article>
```

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

pseudoelements

- Els pseudoelements es comporten de manera similar que els anteriors.
- Les diferències entre pseudoclasses i pseudoelements són:
  - Les pseudoclasses fan referència a la posició de l'element complet en el document, mentre que els pseudoelements fan referència a la posició de determinades parts d'un element en el document.
  - El pseudoelement només pot aparèixer al final del selector, mentre que la pseudoclasse pot aparèixer a qualsevol element del selector.

Sintaxi: Els pseudoelements comencen amb un doble signe de dos punts.

```
::nom del pseudo-element
```

Exemple:

CSS

```
article p::first-line {
   font-size: 14pt;
   font-weight: bold;
}

--------------------------
```

HTML

```
<article>
   <p>Paràgraf al qual canviarem la primera línia, és a dir aplicarem a la primera línia una font de 14pt, i negreta.
</p>
   <p>L'altre paràgraf al qual canviarem la primera línia de la mateixa manera que l'anterior</p>
</article>
```

- Ampliació: *https://developer.mozilla.org/es/docs/Web/CSS/Pseudo-elements*

1.4. Caixes

- **Propietat display**

  Sabem que els elements HTML es divideixen en elements en línia i de blocs. **La propietat display ens permet modificar aquest comportament si ho desitjam. Aquesta propietat és molt important per a controlar estructures**. Els valors més utilitzats són:

  - **"block"**
    - Fa que el comportament de l'element sigui un bloc, és a dir, el seu comportament genera una caixa.
    - Sempre té un poc d'espai per damunt i per davall, que el separa dels demés elements.
    - No accepta cap altre element al seu costat.
    - Per defecte s'expandeix en tot l'ample del seu contenidor.
    - Admet totes les propietat"margin", "padding", "border-width", "border-style", "border", "overflow").
  - **"inline"**
    - El contingut de l'element es posa en línia amb altres elements.
    - Admet: margin, padding, border. No admet: width, height).
  - "**inline-block**"
    - Fa que l'element tengui un comportament mesclat entre bloc i en línia.
  - **"none**"
    - S'utilitza per ocultar un element.
  - **"list-item"**
    - L'element es comporta com un element de llista.

Exemple: Els següents elements són en línia, però amb "display:block" al CSS, podem fer que es comportin com a elements de bloc.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
       span {
          display: block;
          background-color: red;
        }
      a {
         display: block;
         background-color: rgb(196, 196, 240);
       }
    </style>
</head>
<body>
   <span> Text de l'span </span><br>
   <a href="URL"> Text de l'enllaç</a>
</body>
</html>
```

Exemple:Els següents elements són de bloc, però amb "display:inline" al CSS podem fer que es comportin com a elements en línia.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
       h1 {
         display:inline;
         background-color: red;
       }
       p {
         display:inline;
         background-color: rgb(196, 196, 240);
       }
       ul {
         display:inline;
         background-color: rgb(106, 230, 123);
       }
       li {
         display:inline;
         background-color: rgb(213, 221, 99);
       }
    </style>
</head>
<body>
   <h1> Títol </h1>
   <p>Paràgraf</p>
   <ul>
     <li>un</li>
     <li>dos</li>
   </ul>
</body>
</html>
```

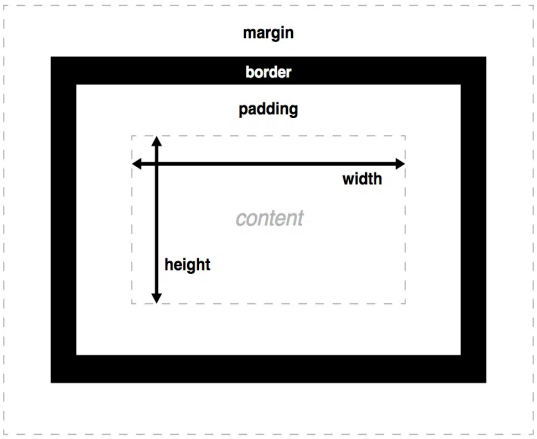
- **Models de caixa**
  - En CSS, en general, hi ha dos tipus de caixes: caixes en bloc i caixes en línia.

    Si una caixa es defineix de tipus "block", es comportarà de la manera següent:

    - Força un salt de línia en arribar al final de la línia.
    - S'estendrà en la direcció de la línia per a omplir tot l'espai disponible que hi hagi en el seu contenidor. En la majoria dels casos, això significa que la caixa serà tan ampla com el seu contenidor, i omplirà el 100% de l'espai disponible.
    - Es respecten les propietats "width" i "height".
    - El farciment, el marge i la vora mantenen als altres elements allunyats de la caixa.

    Si una caixa es defineix externa de tipus "inline", llavors:

    - No força cap salt de línia en arribar al final de la línia.
    - Les propietats width i height no s'apliquen.
    - S'apliquen farcit, marge i vores verticals, però no mantenen allunyades altres caixes en línia.
    - S'apliquen farcit, marge i vores horitzontals, i mantenen allunyades altres caixes en línia.



- **"margin"**: Marge exterior, que separa la caixa de l'element dels altres.

  ```
  Per exemple:
  Si tenim: margin: 5px; establim un marge de 5 píxels als 4 costats de la caixa.
  ```

- **"margin-top", "margin-right", "margin-bottom" i "margin-left"**: Si volem especificar els marges de forma independent.
- **"padding"**: Marge intern o farcit, que s'aplica des del costat cap endins.
- **"padding-top", "padding-right", "padding-bottom" i "padding-left":** Si volem especificar els farcits de forma independent.
- **"border-width"**: Per definir el gruix de la vora.
- **"border-style"**: Per especificar el tipus de línia de la vora.
- **"border"**: Ens permet agrupar ample, estil i color de la vora.
- **"overflow":** Ens permet especificar lo que passa quan el contingut no cap a la caixa, és l'excedent. Els seus valors són:
  - "visible" (Valor predefinit). Encara que el contingut superi la mida de la caixa, es mostrarà tot.
  - "hidden". Només apareix el contingut que no superi el límit de la caixa.
  - "scroll". Apareix una barra de desplaçament.
  - "auto". Es deixa a criteri del navegador el comportament.
- **overflow-x"**: Estableix el desbordament només per l'eix X.
- **"overflow-y"**: Estableix el desbordament només per l'eix Y.

Desplaçaments de caixes

- **top, bottom, left i right**
  - S'utilitzen amb la posició per establir la ubicació d'un element.
  - No es poden aplicar al posicionament estàtic.
  - Aquestes propietats també poden admetre mesures negatives, en aquest cas, la mesura negativa desplaça la caixa en sentit contrari al positiu.

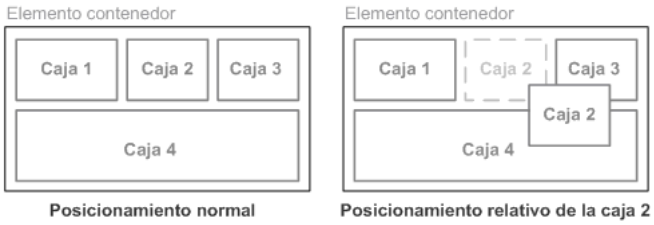
Posicionament de caixes

- **position: static**
  - Els elements es posicionen d'acord a el flux normal de la pàgina.
  - És la posició natural dels elements.
  - Només es té en compte si l'element és de bloc o de línia.
  - No són afectats per les propietats "top", "bottom", "left" i "right".

Exemple:

```
<!DOCTYPE html>
<html>
<head>
    <title>Posicionament</title>
    <meta charset="utf8">
    <style>
    .cont{
        max-width: 500px;
    }
    .caixa{
        position: static;
        width: 200px;
        height: 200px;
        border: 3px solid red;
        color: blue;
    }
    </style>
</head>
<body>
   <div class="cont">
      <a href="" class="caixa">1</a>
      <a href="" class="caixa">2</a>
      <a href="" class="caixa">3</a>
      <a href="" class="caixa">4</a>
   </div>
</body>
</html>
```

- **position: relative**
  - El desplaçament de la caixa es realitza a partir de la seva posició original.
  - La caixa es mou amb les propietats "top", "left", "right" i "bottom".
  - Aquest tipus de posicionament no modifica el posicionament de les demés caixes al seu voltant o caixes adjacents, però si pot provocar solapaments.

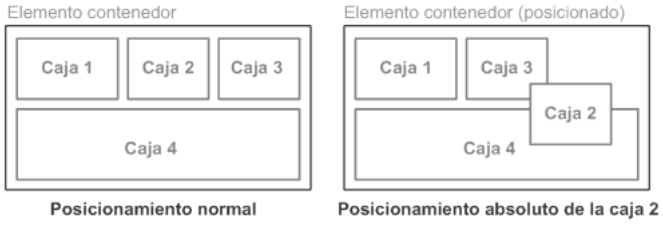


Per exemple:

Si escrivim "left: 15px", desplaçam l'element 15px al seu costat esquerra, de manera podem observar un d esplaçament cap a la dreta.

```
<!DOCTYPE html>
<html>
<head>
    <title>Posicionament</title>
    <meta charset="utf8">
    <style>
    .cont{
        max-width: 500px;
    }
    .caixa{
        position: static;
        width: 200px;
        height: 200px;
        border: 3px solid red;
        color: blue;
    }
    .caixarel{
        position: relative;
        top: 150px;
        left: 150px;
        width: 200px;
        height: 200px;
        color: blue;
        border: 3px solid green;
    </style>
</head>
<body>
   <div class="cont">
      <a href="" class="caixa">1</a>
      <a href="" class="caixa">2</a>
      <a href="" class="caixarel">3</a>
      <a href="" class="caixa">4</a>
   </div>
</body>
</html>
```

- **position: absolute**
  - La posició de partida és sempre des de la vora de la finestra del navegador.
  - El posicionament absolut s'utilitza per a establir de forma precisa la posició en la que es es mostra la caixa d'un element.
  - "left", "top", "right" i "bottom", indiquen la vora des de la qual es mesura el desplaçament, (si és positiu serà sempre cap a l'interior de la finestra).
  - La interpretació dels valors de les propietats de desplaçament ("top", "left", "right" i "bottom"), és un poc complexa, ja que depenen del posicionament de l'element contenidor.



La "caja 2" està posicionada de forma absoluta, el que implica que la resta d'elements ignoren que existeixi,

és per això que la "caja 3", deixa el seu lloc original i passa a ocupar el forat que ha deixat la "caja 2".

  - Per a determinar l'origen de les coordenades a partir del qual es desplaça una caixa posicionada de forma absoluta, podem seguir els següents passos:
    - Cercar tots els contenidors de la caixa fins arribar a l'element "body" de la pàgina.
    - De tots ells, el navegador es queda amb el primer element contenidor que estigui posicionat de qualsevol forma diferent a "static". La cantonada superior esquerra d'aquest element contenidor és l'origen de coordenades.

```
Per exemple:

Si la propietat "top" és 100px, indicarà que la caixa se situarà 100px per davall de la vora superior.
Si la propietat "left" és 200px, indicarà que la caixa se situarà 200px a la dreta de la vora esquerra de la pantalla.
Els valors negatius desplacen la caixa en sentit contrari, per tant quedaran fora de la pantalla, i només serien visibles si la pàgina fos més gran que la finestra.
```

- **position: fixed**
  - És un cas particular del posicionament absolut.
  - Determina la posició d'un element donat a la pàgina HTML. La seva posició és inamovible dins de la finestra del navegador.
  - Inclús si l'usuari utilitza la la barra de desplaçament, aquest element fix no es mourà.
  - Quasi no s'utilitza.

```
Exemple:

<style>
 #fixa {
     position: fixed;
     top: 0px;
     left: 0px;
     border: 1px solid #333;
     background-color: green;
  }
</style>

<body>
    <p id="fixa">La propietat "position:fixed" de CSS </p>
</body>
```

- **position: sticky**
  - Els elements són posicionats de forma relativa fins que el seu bloc contenidor arriba al límit establert.
  - S'usa quan volem que un element tingui una posició relativa fins a un punt i que després canviï a una posició fixa, usant només CSS sense necessitat de codi JavaScript.

```
Per exemple:

Si tenim un banner de publicitat flotant i ens interessa que quan aparegui al fer scroll, es mantingui fix i visible.
```

- **Altres propietats rellevants en el posicionament de caixes**
  - **Propietat float**
    - Quan a un element HTML se li aplica un estil amb la propietat "float", l'element surt del flux normal i apareix posicionat a l'esquerra o a la dreta del seu contenidor, on la resta de elements de la pàgina es posicionaran voltant.
    - Els valors que pot prendre són: "left", "right", "none".
  - **Propietat clear**
    - Estableix si un element ha d'estar a la banda dels elements flotants que el precedeixen o si ha de situar-se davall d'ells.
    - Se sol utilitzar per restaurar el flux normal del document i així els elements deixen de surar cap a l'esquerra, la dreta o ambdós costats.
    - Pot prendre els valors: "none", "left", "right" o "both".
  - **Propietat z-index**
    - Amb la propietat "z-index" seguida d'un valor de nombre sencer podem modificar la superposició de capes, sent els valors numèrics més alts les capes per sobre de les altres.

Per exemple:

```
.primera {
   width: 150px;
   height: 200px;
   background-color: red;
   position: relative;
   z-index: 3;
 }
.segona {
   width: 150px;
   height: 200px;
   background-color: black;
   position: relative;
   left: 50px;
   top: -110px; z-index: 1;
 }
.tercera {
   width: 150px;
   height: 200px;
   background-color: blue;
   position: relative;
   left: 100px;
   top: -210px;
   z-index: 2;
 }
```

```
<html>
<head>
  <meta charset="utf-8">
  <title>z-index</title>
  <link rel="stylesheet" href="estil.css">
</head>
<body>
  <div class="primera"></div>
  <div class="segona"></div>
  <div class="tercera"></div>
</body>
</html>
```

1.5. Cascada i herència

Dues de les **característiques** que fan que els fulls d'estil tenguin una gran potència són:

- La Cascada: La cascada es refereix a la possible combinació de diferents fulls d'estil.
- La Herència: L'herència fa referència a la capacitat que tenen els elements del document HTML d'heretar propietats dels seus elements antecessors.

Els dos conceptes estan relacionats, però com veurem són ben diferents, la cascada té a veure amb les declaracions de CSS que s'apliquen al document i l'herència es refereix a com els elements d'HTML hereten propietats dels seus elements pares i les transmeten als seus fills .

- ***Cascada***

  Les sigles CSS volen dir Fulls d'estil en Cascada. El terme cascada vol dir **que es poden combinar diferents fulls d'estil i que les propietats de tots ells es van acumulant.** Això ens és molt útil quan pensem en llocs web grans, on podem tenir un full d'estil bàsic i anar incorporant-hi altres fulls d'estil, segons les nostres necessitats.

  Però es poden donar alguns conflicte entre declaracions de CSS:

  - Per exemple, imaginem que dins un arxiu CSS extern, hi trobam una declaració: p {font-color: blue;}.
  - Per una altra banda, tenim un CSS intern, amb una declaració com: p {font-color: red;},
  - i també que dins del propi codi HTML trobàssim una declaració com: <p style = "font-color: yellow;".

  Si analitzam, aquestes tres declaracions suposen un conflicte o col·lisió d'estils per el navegador: Quin estil s'ha d'aplicar?
  Per solucionar els conflictes, s'aplica una criteri anomenat de precedència:

  `Declaració en línia > Declaració interna > Declaració externa`

  - Declaració en línia o *Inline style:* Propietats establertes en l'atribut style d'un element.
  - Declaració interna o *Internal Style Sheet:* Propietats establertes en l'element <style> del document (X)HTML.
  - Declaració externa o *External Style Sheet:* Propietats establertes en un full d'estil extern,és la

  D'aquesta manera podem dir que l'ordre de prioritat ve marcat per la proximitat amb l'element que l'afecta, sent la més prioritària Inline Style i la menys External Style Sheet.

  Pot passar també que els navegadors permetin a l'usuari establir diferents propietats d'estil: mida de la lletra, colors, etc., en aquest cas, són aquestes les que tenen més prevalença.

  Exemple 1.

  Tenim tres declaracions:

  El fitxer index.html:

```
<!doctype html >
<html lang="ca">
<head>
...
<link rel="stylesheet" href="estils.css" />
<style>
        p {
                color: red;
        }
</style>
</head>
<body>
        <h1>Encapçalament</h1>
        <p style="color: blue">Un paràgraf</p>
        <p>Un altre paràgraf</p>
</body>
</html>
```

  El fitxer estils.css:

```
p {
        color: green;
}
h1 {
        color: violet;
}
```

  - El full d'estil "estils.css", té una regla que ens diu que tots els paràgrafs tindran la lletra de color verd.
  - L'element *style* del document "index.html", ens diu que tots els paràgrafs d'aquest document seran de color vermell.
  - L'etiqueta <style> del primer paràgraf del document "index.html", ens diu que aquest paràgraf serà de color blau.

  De quin color seran els paràgrafs? La resposta és:

  - El primer paràgraf serà de color vermell, perquè l'atribut "style" és la ubicació amb més alta jerarquia.
  - El segon paràgraf serà de color verd, ja que no té cap <style>.
  - Per últim, el color de lletra del títol <h1> serà violeta, tal com està establert al fitxer "estils.css", ja que és l'única regla que tenim que faci referència a l'etiqueta <h1>.

- ***Herència***

  Tots els elements d'una pàgina HTML, amb l'excepció de l'element arrel <html>, estan continguts en un altre element. L'herència és que tot element hereta les propietats dels seus elements antecessors.

S'ha de tenir en compte el següent:

- No totes les propietats s'hereten. Aquesta característica està descrita en l'especificació de CSS corresponent.
- Si volem forçar l'herència en una propietat, podem introduir-hi el valor inherit:
  - inherit és un valor permès a totes les propietats CSS. Fa que l'element al qual s'aplica, prengui el valor calculat de la propietat del seu element pare.
  - Per a propietats heretades: Aquest valor reforça el comportament del valor per defecte i és necessari per anular altres regles. Per exemple:

```
  /* Per fer les capçaleres de segon nivell, de color verd */
  h2 { color: green; }
  /* ...però deixa les capçaleres dins de la barra lateral amb el valor per defecte (el valor que tengui l'elemen
t pare */
  #sidebar h2 { color: inherit; }
```

  - Per a propietats no heretades: Aquest valor especifica un comportament que té relativament poc sentit, al no reforçar el valor per defecte.
- Si posem un valor a una propietat, aquest valor preval sobre el valor heretat.
- Els elements hereten el valor computat, no el valor especificat. És a dir, si una propietat té per valor un valor relatiu (per exemple, un percentatge), el valor heretat és el resultat calculat.

Exemple:

Tenim el següent document HTML:

```
<<! DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3c.org/TR/xhtml1/DTD/
xhtml1-strict.dtd " >
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="ca" lang="ca">
<head>
  ...
  <style>
    body { font-size: small; }
    p { color: gray; }
    strong { font-style: italic; }
  </style>
</head>
<body>
    <p>Un paràgraf amb una <strong>text molt emfatitzat</strong></p>
</body>
</html>
```

L'element és a dins de l'element <p>, que està contingut a l'element <body>, això vol dir que la cadena text molt emfatitzat tindrà les següents propietats:

- Estarà en cursiva perquè així ho especifica la propietat font-style: italic de tots els elements ,
- Com que aquest element és a dins de l'element <p> i el color de lletra de tots els paràgrafs és el gris, tal com diu la propietat color: gray, la cadena també serà de color gris.
- És més, com que l'element <p> és a dins de l'element <body>, també hereta la declaració font-size: small.
- En resum, la cadena text molt emfatitzat tindrà lletra petita, serà de color gris i estarà en cursiva.