# DESENVOLUPAMENT D'APLICACIONS WEB
# M09 - Disseny d'interfícies web
# UF1 - Disseny de la interfície

## Fulls d'estil en cascada (CSS)

---

## 1. Introducció

### 1.1. Què és CSS?

CSS és l'acrònim de *Cascading Style Sheets* (fulls d'estil en cascada). Es va desenvolupar amb l'objectiu de crear una especificació que separés els contingut de la pàgina Web HTML de la presentació d'aquesta amb l'estil CSS, facilitant així el treball a l'hora de crear pàgines Webs.

Els estils defineixen en fitxers CSS (*.css) com mostrar els elements HTML/XHTML, es a dir, li donen el format de forma separada al contingut, estalviant d'aquesta manera molta feina. La versió estàndard actualment és la 2.1.

La primera versió es va llançar l'any 1996 i va ser elaborada per **Hakon Wium Lie** i **Bert Bos**, que posteriorment van ser els creadors del navegador Opera, que si bé mai ha estat un navegador molt estès, va tenir molta influència sobre la resta de navegadors.

CSS va aparèixer inicialment per posar una mica d'ordre al desordre que hi havia a l'hora d'aplicar estils en els diferents navegadors. Fins que no va sortir CSS, cada navegador donava estils a les pàgines web d'una forma diferent.

S'ha de tenir en compte que cada navegador té el seu motor de renderització i depenen d'aquest hi hauran petites diferències en la interpretació que el navegador fa de l'estil:

- Blink (Chrome i Opera). (és un *fork* de Webkit)
- Gecko (Firefox)
- Webkit (Safari)
- EdgeHTML (Edge)
- KHTML (Konqueror)

La seva primera versió va ser limitada, ja que només aplicava estils visuals al tipus de lletra, colors al text i al fons, permetia alinear el text juntament amb les imatges i les taules, es podia assignar marge, vores, *padding* i posicionar de certa manera els elements.

Conforme han anat avançant els anys, van aparèixer noves propietats i noves millores, fins a arribar a la versió que tenim actualment de CSS3.

### 1.2. Avantatges de CSS

HTML / XHTML és un llenguatge òptim per a estructurar i crear continguts, **no pas per aplicar formats o crear estils**.

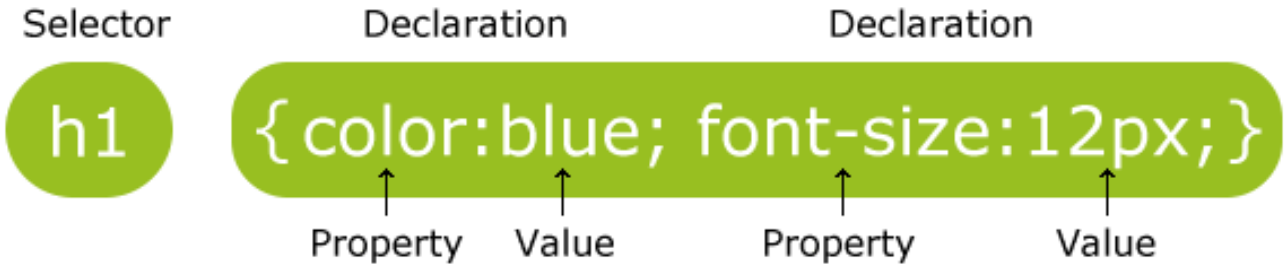
Afegir el format d'una pàgina dins el codi HTML genera fitxers *.html molt complexos i amb molta informació i per tant amb més probabilitat de contenir errors. A més son més lents de llegir pels navegadors.

Si volem canviar només el contingut de la pàgina mantenint l'estil, i tenim el codi que dona el format i el contingut junts, ho hem de reescriure tot. Si per contra, volem canviar l'estil sense canviar el contingut també ho hem de canviar tot.

Amb CSS podem canviar l'estil o el contingut de forma independent, sense fer canvis o amb petits canvis.

### 1.3. Sintaxi

La sintaxi de les regles a CSS té els següents elements:




| | |
|---|---|
| **selector** (selector) | connecta el contingut a un estil determinat |
| **declaració** (declaration) | és la instrucció, està formada per propietat/s + valor/s |
| **propietat** (property) | fa referència a la característica a modificar (color, posició, ...) |
| **valor** (value) | el grau o tipus d'estil (en color seria: red, blue, green, ...) |

A més i haurien els separadors:

| | |
|---|---|
| { } | al començament i final de la/es declaració/ons |
| **:** | entre la propietat i el valor |
| **;** | al final del valor |
| **,** | per combinar selectors |

### 1.4. Creació i vinclament de fulls d'estil

La definició d'un estil amb CSS ha de està vinculat amb la pàgina amb el codi HTML per que s'apliqui. Hi ha fonamentalment tres sistemes per fer-ho:

#### En línia

Dins del mateix fitxer que l'HTML abans del contingut que ha de donar forma.

```
<div style="font-size:18px; font-family:arial; color:red;">Lorem
ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
temporincididunt ut labore et dolore magna aliqua.</div>
```

#### Dins la capçalera

Dins de <head></head>. Concretament ha d'anar dins de l'etiqueta <style> </style>.

```
!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=UTF-8"/>
<title></title>
<style type="text/css">
p { color: black; font-family: Verdana; font-size:18px;}
</style>
</head>
<body>
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua.</p>
</body>
</html>
```

#### A un fitxer CSS extern

Enllaçat des de la pàgina HTML.

```
<link href="CSS1.css" rel="stylesheet" type="text/css"/>
```

## 2. Selectors bàsics

Una regla de CSS està formada pel "selector" i la "declaració".

La declaració indica **què cal fer** i el selector indica **on cal fer-li-ho**.

A un mateix element HTML se li poden aplicar diverses regles CSS i cada regla CSS pot aplicar-se a un nombre il·limitat d'elements.

CSS té una dotzena de tipus diferents de selectors, que permeten seleccionar de forma molt precisa elements individuals o conjunts d'elements dins d'una pàgina web, tot i que la majoria de pàgines web es poden dissenyar utilitzant els cinc selectors bàsics:

- Selector universal
- Selector de tipus o etiqueta
- Selector descendent
- Selector de classe
- Selector d'ID

### 2.1. Selector universal

S'utilitza per seleccionar tots els elements de la pàgina. El següent exemple elimina el marge i l'emplenament de tots els elements:

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

El selector universal s'indica mitjançant un asterisc (*). Malgrat la seva senzillesa, no s'utilitza gaire, ja que és difícil que un mateix estil es pugui aplicar a tots els elements d'una pàgina.

### 2.2. Selector de tipus o etiqueta

Selecciona tots els elements de la pàgina l'etiqueta de la qual HTML coincideix amb el valor del selector. El següent exemple selecciona tots els paràgrafs de la pàgina:

```
p {
...
}
```

Per utilitzar aquest selector, només cal indicar el nom d'una etiqueta HTML (sense els caràcters < i >) corresponent als elements que es volen seleccionar.

El següent exemple aplica diferents estils als titulars i als paràgrafs d'una pàgina HTML:

```
h1 {
color: red;
}
```

```
h2 {
color: blue;
}

p {
color: black;
}
```

Si es vol aplicar els mateixos estils a dues etiquetes diferents, es poden encadenar els selectors.

En el següent exemple, els títols de secció **h1**, **h2** i **h3** comparteixen els mateixos estils:

```
h1 {
color: #8A8I27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}

h2 {
color: #8A8I27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}

h3 {
color: #8A8I27;
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. Per aquesta raó s'inclouen tots els selectors separats per una coma (,) i el resultat és que la següent regla CSS és equivalent a les tres regles anteriors:

```
h1, h2, h3 {
color: #8A8I27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}
```

En els fulls d'estil complexes, és habitual agrupar les propietats comunes de diversos elements en una única regla CSS i posteriorment definir les propietats específiques d'aquests mateixos elements.

El següent exemple estableix en primer lloc les propietats comunes dels títols de secció (color i tipus de lletra) i a continuació, estableix la grandària de lletra de cadascun d'ells:

```
h1, h2, h3 {
color: #8A8I27;
font-weight: normal;
font-family: Arial, Helvetica, sans-serif;
}
```

```
h1 { font-size: 2em; }
h2 { font-size: 1.5em; }
h3 { font-size: 1.2em; }
```

### 2.3. Selector descendent

Selecciona els elements que es troben dins d'altres elements. Un element és descendent d'un altre quan es troba entre les etiquetes d'obertura i de tancament de l'altre element.

El selector del següent exemple selecciona tots els elements de la pàgina que es trobin dins d'un element **p**:

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

Si el codi HTML de la pàgina és el següent:

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

El selector **p span** selecciona tant **text1** com a **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 importar el nivell de profunditat en el qual es trobi.

A la resta d'elements **** de la pàgina que no estan dins d'un element no se'ls aplica la regla CSS anterior.

Els selectors descendents permeten augmentar la precisió del selector de tipus o etiqueta. Així, utilitzant el selector descendent és possible aplicar diferents estils als elements del mateix tipus. El següent exemple amplia l'anterior i mostra de color blau tot el text dels **** continguts dins d'un **<h1>**:

```
p span { color: red; }
h1 span { color: blue; }
```

Amb les regles CSS anteriors:

- Els elements **** que es troben dins d'un element **<p>** es mostren de color vermell.
- Els elements **** que es troben dins d'un element **<h1>** es mostren de color blau.
- La resta d'elements de la pàgina, es mostren amb el color per defecte aplicat pel navegador.

La sintaxi del selector descendent es mostra a continuació:

```
selector1 selector2 selector3 ... selectorN
```

Els selectors descendents sempre estan formats per dos o més selectors separats entre si per espais en blanc. L'últim selector indica l'element sobre el qual s'apliquen els estils i tots els selectors anteriors indiquen el lloc en el qual s'ha de trobar aquest element.

En el següent exemple, el selector descendent es compon de quatre selectors:

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

Els estils de la regla anterior s'apliquen als elements de tipus ****, que a l'hora es trobin dins d'elements de tipus **<a>** que es trobin dins d'elements de tipus **<p>**.

### 2.4. Selector de classe

Si es considera el següent codi HTML d'exemple:

```
<div>
     <p>Lorem ipsum dolor sit amet...</p>
     <p>Nunc sed lacus et est adipiscing accumsan...</p>
     <p>Class aptent taciti sociosqu ad litora...</p>
</div>
```

Com es poden aplicar estils CSS només al primer paràgraf?

- El selector universal (*****) no es pot utilitzar perquè selecciona tots els elements de la pàgina.
- El selector de tipus o etiqueta **p** tampoc es pot utilitzar perquè seleccionaria tots els paràgrafs.
- Finalment, el selector descendent (**div p**) tampoc es pot utilitzar perquè tots els paràgrafs es troben en el mateix lloc.

Una de les solucions més senzilles per aplicar estils a un sol element de la pàgina consisteix a utilitzar l'atribut **class** d'HTML sobre aquest element per indicar directament la regla CSS que se li ha d'aplicar:

```
<div>
     <p class="destacat">Lorem ipsum dolor sit amet...</p>
     <p>Nunc sed lacus et est adipiscing accumsan...</p>
     <p>Class aptent taciti sociosqu ad litora...</p>
</div>
```

A continuació, es crea en l'arxiu CSS una nova regla anomenada **destacat** amb tots els estils que es van a aplicar a l'element. Perquè el navegador no confongui aquest selector amb els altres tipus de selectors, es prefixa el valor de l'atribut **class** amb un punt (**.**) tal com apareix al següent exemple:

```
.destacat { color: red; }
```

Aquest selector rep el nom de selector de **classe** i és un dels més emprats. La principal característica d'aquest selector és que en una mateixa pàgina HTML elements diferents poden fer servir el mateix valor en l'atribut **class**:

Aquests selectors permeten reutilitzar els mateixos estils per a diversos elements diferents com es pot apreciar als següents exemples:

```
.avis {
padding: 0.5em;
border: 1px solid #98be10;
background: #f6feda;
 }

.error {
color: #930;
font-weight: bold;
}

<span class="error">...</span>

<div class="avis">...</div>
```

L'element té un atribut **class="error"**, així que se li apliquen les regles CSS indicades pel selector **.error**. Per la seva banda, l'element té un atribut **class="avis"**, raó per la que el seu estil és el que defineixen les regles CSS del selector **.avis**.

En ocasions és necessari restringir l'abast del selector de classe. Si es considera de nou l'exemple anterior:

```
<div>
     <p class="destacat">Lorem ipsum dolor sit amet...</p>
     <p>Nunc sed lacus et <a href="#" class="destacat">est
          adipiscing</a> accumsan...</p>
     <p>Class aptent taciti <em class="destacat">sociosqu ad</em>
          litora...</p>
</div>
```

Com és possible aplicar estils només al paràgraf l'atribut del qual class sigui igual a destacat?

Combinant el selector de tipus i el selector de classe, s'obté un selector molt més específic:

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

El selector **p.destacat** s'interpreta com "aquells elements de tipus que disposin d'un atribut **class** amb valor **destacat**". De la mateixa forma, el selector **a.destacat** només selecciona els enllaços on l'atribut **class** sigui igual a **destacat**.

De l'anterior es dedueix que l'atribut **.destacat** és equivalent a ***.destacat**, per la qual cosa s'obvia el símbol * en escriure un selector de classe normal.

No s'ha de confondre el selector de classe amb els selectors anteriors:

```
/* Tots els elements de tipus "p" amb atribut class="avis" */
p.avis { ... }

/* Tots els elements amb atribut class="avis" que estiguin dins de
qualsevol element de tipus "p" */
p .avis { ... }

/* Tots els elements "p" de la pàgina i tots els elements amb
atribut class="avis" de la pàgina */
p, .avis { ... }
```

Finalment, és possible aplicar els estils de diverses classes CSS sobre un mateix element. La sintaxi és similar, però els diferents valors de l'atribut **class** se separen amb espais en blanc. En el següent exemple:

```
<p class="especial destacat error">Paràgraf de text...</p>
```

Al paràgraf anterior se li apliquen els estils definits en les regles **.especial**, **.destacat** i **.error**, per la qual cosa en el següent exemple, el text del paràgraf es veuria de color vermell, en negreta i amb una grandària de lletra de 15 píxel:

```
.error { color: red; }
.destacat { font-size: 15px; }
.especial { font-weight: bold; }

<p class="especial destacado error">Paràgraf de text ...</p>
```

Si un element disposa d'un atribut **class** amb més d'un valor, és possible utilitzar un selector més avançat:

```
.error { color: red; }
.error.destacat { color: blue; }
.destacat { font-size: 15px; }
.especial  { font-weight: bold; }

<p class="especial destacat error">Paràgraf de text...</p>
```

En l'exemple anterior, el color de la lletra del text és blau i no vermell. El motiu és que s'ha utilitzat un selector de classe múltiple **.error.destacat**, que s'interpreta com "aquells elements de la pàgina que disposin d'un atribut **class** amb almenys els valors **error** i **destacat**".

### 2.5. Selectors d'ID

En ocasions, és necessari aplicar estils CSS a un únic element de la pàgina. Encara que pot utilitzar-se un selector de classe per aplicar estils a un únic element, existeix un altre selector més eficient en aquest cas.

El selector d'ID permet seleccionar un element de la pàgina a través del valor del seu atribut **ID**. Aquest tipus de selectors només seleccionen un element de la pàgina perquè

el valor no es pot repetir en dos elements diferents d'una mateixa pàgina.

La sintaxi dels selectors d'ID és molt semblant a la dels selectors de classe, tret que s'utilitza el símbol del coixinet (#) en comptes del punt (.) com a prefix del nom de la regla CSS:

```
#destacat { color: red; }

<p>Primer paràgraf</p>
<p id="destacat">Segon paràgraf</p>
<p>Tercer paràgraf</p>
```

En l'exemple anterior, el selector **#destacat** només selecciona el segon paràgraf.

La principal diferència entre aquest tipus de selector i el selector de classe té a veure amb HTML i no amb CSS. Com se sap, en una mateixa pàgina, el valor de l'atribut **ID** ha ser únic, de manera que dos elements diferents no poden tenir el mateix valor d'**ID**. No obstant això, l'atribut **class** no és obligatori que sigui únic, de manera que molts elements HTML diferents poden compartir el mateix valor per al seu atribut **class**.

La recomanació general és la d'utilitzar el selector d'**ID** quan es vol aplicar un estil a un sol element específic de la pàgina i utilitzar el selector de classe quan es vol aplicar un estil a diversos elements diferents de la pàgina HTML.

Igual que els selectors de classe, en aquest cas també es pot restringir l'abast del selector mitjançant la combinació amb altres selectors. El següent exemple aplica la regla CSS només a l'element de tipus <p> que tingui un atribut **ID** igual a l'indicat:

```
p#aviso { color: blue; }
```

A primera vista, restringir l'abast d'un selector d'**ID** pot semblar absurd. En realitat, un selector de tipus **p#avis** només té sentit quan l'arxiu CSS s'aplica sobre moltes pàgines HTML diferents.

En aquest cas, algunes pàgines poden disposar d'elements amb un atribut **ID** igual a **avis** que no siguin paràgrafs, per la qual cosa la regla anterior no s'aplica sobre aquests elements.

No s'ha de confondre el selector d'ID amb els selectors anteriors:

```
/* Tots els elements de tipus "p" amb atribut ID="avis" */
p#avis { ... }

/* Tots els elements amb atribut ID="avís" que estiguin dins de
qualsevol element de tipus "p" */
p #avis { ... }

/* Tots els elements "p" de la pàgina i tots els elements amb
atribut aneu="avís" de la pàgina */
p, #avís { ... }
```

### 2.6. Combinació de selectors bàsics

CSS permet la combinació d'un o més tipus de selectors per restringir l'abast de les regles CSS. A continuació es mostren alguns exemples habituals de combinació de selectors.

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

L'anterior selector només selecciona aquells elements amb un class="especial" que es trobin dins de qualsevol element amb un class="avis".

Si es modifica l'anterior selector:

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

Ara, el selector només selecciona aquells elements de tipus amb un atribut class="especial" que estiguin dins de qualsevol element de tipus que tingui un atribut class="avis".

### 3. Agrupaments de regles

Quan es creen arxius CSS complexos amb desenes o centenars de regles, és habitual que els estils que s'apliquen a un mateix selector es defineixin en diferents regles:

```
h1 { color: red; }
...
h1 { font-size: 2em; }
...
h1 { font-family: Verdana; }
```

Les tres regles anteriors estableixen el valor de tres propietats diferents dels elements. Abans que el navegador mostri la pàgina, processa totes les regles CSS de la pàgina per tenir en compte tots els estils definits per a cada element.

Quan el selector de dos o més regles CSS és idèntic, es poden agrupar les declaracions de les regles per fer les fulls d'estils més eficients:

```
h1 {
color: red;
font-size: 2em;
font-family: Verdana;
}
```

L'exemple anterior té el mateix efecte que les tres regles anteriors, però és més eficient i és més fàcil de modificar i mantenir per part dels dissenyadors.

Si es vol reduir al màxim la grandària de l'arxiu CSS per millorar lleugerament el temps de càrrega de la pàgina web, també és possible indicar la regla anterior de la següent forma:

```
h1 {color:red;font-size:2em;font-family:Verdana;}
```

## 4. Herència

Una de les característiques principals de CSS és l'herència dels estils definits per als elements. Quan s'estableix el valor d'una propietat CSS en un element, els seus elements descendents hereten de forma automàtica el valor d'aquesta propietat.

Si es considera el següent exemple :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
<title>Herencia d'estils</title>
<style type="text/css">
body { color: blue; }
</style>
</head>

<body>
<h1>Titular de la pàgina</h1>
<p>Paràgraf de text no gaire llarg.</p>
</body>
</html>
```

A l'exemple anterior, el selector **body** només estableix el color de la lletra per a l'element **<body>**. La propietat color és una de les que s'hereten de forma automàtica, raó per la que tots els elements descendents de **<body>** es representen amb el mateix color de lletra. Així, doncs, definint el color de la lletra al element **<body>** canvia el color de lletra de tots els elements de la pàgina.

Encara que l'herència d'estils s'aplica automàticament, es pot anul·lar el seu efecte establint de forma explícita un altre valor per a la propietat que s'hereta, com es mostra en el següent exemple:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejemplo de herencia de estilos</title>
<style type="text/css">
body { font-family: Arial; color: black; }
h1 { font-family: Verdana; }
p { color: red; }
</style>
</head>

<body>
<h1>Titular de la pàgina</h1>
<p>Un paràgraf de text.</p>
</body>
</html>
```

A l'exemple anterior, es defineix en primer lloc el color i tipus de lletra de l'element **<body>**, així que els altres elements heretarien aquestes característiques si no fos per que les altres regles modifiquen algun dels estils heretats.

D'aquesta forma, els elements **<h1>** de la pàgina es mostren amb el tipus de lletra **Verdana** establert pel selector **<h1>** i es mostren de color negre que és el valor heretat de l'element **<body>**.

de la pàgina es mostren del color vermell establert pel selector p i amb un tipus de lletra Arial heretat de l'element

La majoria de propietats CSS apliquen l'herència d'estils de forma automàtica. A més, per a aquelles propietats que no s'hereten automàticament, CSS inclou un mecanisme per forçar al fet que s'heretin els seus valors.

Finalment, tot i que l'herència automàtica d'estils pot semblar complicada, simplifica en gran manera la creació de fulls d'estils complexes. Com s'ha vist en els exemples anteriors, si es vol establir la tipografia base de la pàgina, simplement s'ha d'establir en l'element **<body>** de la pàgina i la resta dels elements l'heretaran.

### 5. Col·lisions d'estils

En les fulls d'estils complexes, és habitual que diverses regles CSS s'apliquin a un mateix element HTML. El problema d'aquestes regles múltiples és que es poden donar col·lisions com la del següent exemple:

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

<p> ... </p>
```

De quin color es mostra el paràgraf anterior? CSS té un mecanisme de resolució de col·lisions molt complex i que té en compte el tipus de full d'estil que es tracti (de navegador, d'usuari o de dissenyador), la importància de cada regla i l'específic que sigui el selector.

El mètode seguit per CSS per resoldre les col·lisions d'estils es mostra a continuació:

1. Determinar totes les declaracions que s'apliquen a l'element per al mitjà CSS seleccionat.
2. Ordenar les declaracions segons el seu origen (CSS de navegador, d'usuari o de dissenyador) i la seva prioritat.
3. Ordenar les declaracions segons l'específic que sigui el selector. Com més genèric és un selector, menys importància tenen les seves declaracions.
4. Si després d'aplicar les normes anteriors existeixen dues o més regles amb la mateixa prioritat, s'aplica la que es va indicar en últim lloc.

Tot i que és més complicat, de forma simplificada es pot fer servir és el següent:

- Com més específic sigui un selector, més importància té la seva regla associada.
- A igual especificitat, es considera l'última regla indicada.

Com a l'exemple anterior els dos selectors són idèntics, les dues regles tenen la mateixa prioritat i preval la que es va indicar en últim lloc, per la qual cosa el paràgraf es mostra de color blau.

En el següent exemple, la regla CSS que preval es decideix per l'específic que és cada selector:

```
* { color: blue; }
p { color: red; }
p#especial { color: green; }

<p id="especial">...</p>
```

A l'element **<p>** se li apliquen les tres declaracions. Com el seu origen i la seva importància és la mateixa, decideix l'especificitat del selector.

- El selector ***** és el menys específic, ja que es refereix a "tots els elements de la

pàgina".

- El selector **p** és poc específic perquè es refereix a "tots els paràgrafs de la pàgina".
- El selector **p#especial** només fa referència a "el paràgraf de la pàgina l'atribut de la qual **id** sigui igual a especial".

Com que el selector **p#especial** és el més específic, la seva declaració és la que es té en compte i per tant el paràgraf es mostra de color verd.

## 6. Unitats de mida

Les mides en CSS s'empren per definir característiques como l'alçària, amplària i marges dels elements i per establir la grandària de lletra del text. Totes les mides s'indiquen com un valor numèric sencer o decimal seguit d'una unitat de mida (sense cap espai en blanc entre el nombre i la unitat de mida).

CSS divideix les unitats de mida en dos grups: absolutes i relatives.

Les **unitats relatives** defineixen el seu valor en relació amb una altra mida, així, per obtenir el seu valor real, s'ha de realitzar alguna operació amb el valor indicat. Les **unitats absolutes** estableixen de forma totalment definida el valor d'una mida, per la qual cosa el seu valor real és directament el valor indicat.

Si el valor és 0, la unitat de mida és opcional. Si el valor és diferent a 0 i no s'indica cap unitat, la mida s'ignora completament.

Algunes propietats permeten indicar mides negatives, encara que habitualment els seus valors són positius. Si el valor decimal d'una mida és inferior a 1, es pot ometre el 0 de l'esquerra (0.5em és equivalent a .5em).

### 6.1. Unitats absolutes

Una mida indicada mitjançant unitats absolutes està completament definida, ja que el seu valor no depèn d'un altre valor de referència. A continuació es mostra la llista completa d'unitats absolutes definides per CSS i el seu significat:

- **in**, polzades ("inches", en anglès). Una polzada equival a 2.54 centímetres
- **cm**, centímetres
- **mm**, mil·límetres
- **pt**, punts. Un punt equival a 1 polzada/72, és a dir, uns 0.35 mil·límetres
- **pc**, piques. Una pica equival a 12 punts, és a dir, uns 4.23 mil·límetres.

A continuació es mostren exemples d'utilització d'unitats absolutes:

```
/* El cos de la pàgina ha de mostrar un marge de mitja polzada */
body { margin: 0.5in; }

/* Els elements <h1> han de mostrar un interlineat de 2 centímetres */
h1 { line-height: 2cm; }

/* Les paraules de tots els paràgrafs han d'estar separades 4 mil·límetres
entre si    */
p { word-spacing: 4mm; }

/* Els enllaços s'han de mostrar amb una grandària de lletra de 12 punts
*/
a { font-size: 12pt }
```

```
/* Els elements han de tenir una grandària de lletra d'1 pica */
span { font-size: 1pc }
```

El principal avantatge de les unitats absolutes és que el seu valor és directament el valor que s'ha d'utilitzar, sense cap càlcul afegit. El seu principal desavantatge és que són molt poc flexibles i no s'adapten fàcilment als diferents mitjans.

De totes les unitats absolutes, l'única que sol utilitzar-se és el punt (pt). Es tracta de la unitat de mida preferida per establir la grandària del text en els documents que es van a imprimir, és a dir, per al mitjà print de CSS, tal com es veurà més endavant.

### 6.2. Unitats relatives

La unitats relatives, a diferència de les absolutes, no estan completament definides, ja que el seu valor sempre està referenciat respecte a un altre valor. Malgrat la seva aparent dificultat, són les més utilitzades en el disseny web per la flexibilitat amb la qual s'adapten als diferents mitjans.

A continuació es mostren les tres unitats de mida relatives definides per CSS i la referència que pren cadascuna per determinar el seu valor real:

- **em**, (no confondre amb l'etiqueta *d'HTML) relativa respecte de la grandària de lletra de l'element*
- **ex**, relativa respecte de l'altura de la lletra x ("ics minúscula") del tipus i grandària de lletra de l'element
- **px**, (píxel) relativa respecte de la resolució de la pantalla del dispositiu en el qual es visualitza la pàgina HTML.

Les unitats **em** i **ex** no han estat creades per CSS, sinó que porten dècades utilitzant-se en el camp de la tipografia. Encara que no és una definició exacta, la unitat 1em equival a l'amplària de la lletra M ("ema majúscula") del tipus i grandària de lletra de l'element.

La unitat em fa referència a la grandària en punts de la lletra que s'està utilitzant. Si s'utilitza una tipografia de 12 punts, 1em equival a 12 punts. El valor de 1ex es pot aproximar per 0.5 em.

Si es considera el següent exemple:

```
p { margin: 1em; }
```

La regla CSS anterior indica que els paràgrafs han de mostrar un marge d'amplària igual a 1em. Com es tracta d'una unitat de mida relativa, és necessari realitzar un càlcul matemàtic per determinar l'amplària real d'aquest marge.

La unitat de mida em sempre fa referència a la grandària de lletra de l'element. D'altra

banda, tots els navegadors mostren per defecte el text dels paràgrafs amb una grandària de lletra de 16 píxel. Per tant, en aquest cas el marge de 1em equival a un marge d'amplària 16px.

A continuació es modifica l'exemple anterior per canviar la grandària de lletra dels paràgrafs:

```
p { font-size: 32px; margin: 1em; }
```

El valor del marge segueix sent el mateix en unitats relatives (1em) però el seu valor real ha variat perquè la grandària de lletra dels paràgrafs ha variat. En aquest cas, el marge tindrà una amplària de 32px, ja que 1em sempre equival a la grandària de lletra de l'element.

Si es vol reduir l'amplària del marge a 16px però mantenint la grandària de lletra dels paràgrafs en 32px, s'ha d'utilitzar la següent regla CSS:

```
p { font-size: 32px; margin: 0.5em; }
```

El valor 0.5em s'interpreta com "la meitat de la grandària de lletra de l'element", ja que s'ha de multiplicar per 0.5 la seva grandària de lletra (32px x 0.5 = 16px). De la mateixa forma, si es vol mostrar un marge de 8px d'amplària, s'hauria d'utilitzar el valor 0.25em, ja que 32px x 0.25 = 8px.

El gran avantatge de les unitats relatives és que sempre mantenen les proporcions del disseny de la pàgina. Establir el marge d'un element amb el valor 1em equival a indicar que "el marge de l'element ha de ser de la mateixa grandària que la seva lletra i ha de canviar proporcionalment".

En efecte, si la grandària de lletra d'un element augmenta fins a un valor enorme, el seu marge de 1em també serà enorme. Si la seva grandària de lletra es redueix fins a un valor diminut, el marge de 1em també serà diminut. L'ús d'unitats relatives permet mantenir les proporcions del disseny quan es modifica la grandària de lletra de la pàgina.

El funcionament de la unitat ex és idèntic a em, tret que en aquest cas, la referència és l'altura de la lletra x minúscula, per la qual cosa el seu valor és aproximadament la meitat que el de la unitat em.

Finalment, les mides indicades en píxel també es consideren relatives, ja que l'aspecte dels elements dependrà de la resolució del dispositiu en el qual es visualitza la pàgina HTML. Si un element té una amplària de 400px, ocuparà la meitat d'una pantalla amb una resolució de 800x600, però ocuparà menys de la tercera part en una pantalla amb resolució de 1440x900.

Les unitats de mida es poden barrejar en els diferents elements d'una mateixa pàgina, com en el següent exemple:

```
body { font-size: 10px; }
h1 { font-size: 2.5em; }
```

En primer lloc, s'estableix una grandària de lletra basi de 10 píxel per a tota la pàgina. A

continuació, s'assigna una grandària de 2.5em a l'element <h1>, per la qual cosa la seva grandària de lletra real serà de 2.5 x 10px = 25px.

El valor de les mides relatives no s'hereta directament, sinó que s'hereta el seu valor real una vegada calculat. El següent exemple mostra aquest comportament:

```
bodY {
font-size: 12px;
text-indent: 3em;
}

h1 { font-size: 15px }
```

La propietat **text-indent** s'utilitza per tabular la primera línia d'un text. L'element **<body>** no ho fa, per la qual cosa heretarà el valor del seu element pare. No obstant això, el valor heretat no és **3em**, sinó **36px**.

Si s'heretés el valor **3em**, en multiplicar-ho pel valor de **font-size** de l'element **<h1>** (que val **15px**) el resultat seria **3em x 15px = 45px**. No obstant això, com s'ha comentat, els valors que s'hereten no són els relatius, sinó els valors ja calculats.

Per tant, en primer lloc es calcula el valor real de **3em** per a l'element **3em x 12px = 36px**. Una vegada calculat el valor real, est és el valor que s'hereta per a la resta d'elements.

### Percentatges

El percentatge també és una unitat de mida relativa, encara que per la seva importància CSS la tracta de forma separada a **em**, **ex** i **px**. Un percentatge està format per un valor numèric seguit del símbol **%** i sempre està referenciat a una altra mida. Cadascuna de les propietats de CSS que permeten indicar com a valor un percentatge, defineix el valor al que fa referència aquest percentatge.

Els percentatges es poden utilitzar per exemple per establir el valor de la grandària de lletra dels elements:

```
body { font-size: 1em; }
h1 { font-size: 200%; }
h2 { font-size: 150%; }
```

Les grandàries establertes per als elements **<h1>** i **<h2>** mitjançant les regles anteriors, són equivalents a **2em** i **1.5em** respectivament, per la qual cosa és més habitual definir-los mitjançant **em**.

Els percentatges també s'utilitzen per establir l'amplària dels elements:

```
div#contingut { width: 600px; }
div.principal { width: 80%; }

<div id="contenido">
      <div class="principal">
```

```
        ...
        </div>
    </div>
```

A l'exemple anterior, la referència del valor 80% és l'amplària del seu element pare. Per tant, l'element **<h1>** l'atribut del qual **class** val principal té una amplària de **80% x 600px = 480px**.

## 7. Color

Existeixen diverses formes donar color als elements, cadascuna té els seus avantatges i els seus inconvenients.

### 7.1. Nom

Hi ha dissenys que no són molt complexos, llavors hi hauria prou per a donar color afegir el nom del color (en anglès) en qüestió a la propietat. Aquest sistema només serveix pels colors principals. És forma més simple d'aplicar color a un element.

```
p {
color: lime;
background-color: red;
}
```

### 7.2. RGB hexadecimal

Aquesta forma d'implementar valors és més precisa que la primera, i està destinada a aconseguir major precisió en el color que volem definir.

Utilitza els valors hexadecimals (*base 16*), que fan servir els dígits del 0 al 9 i les lletres de la a a la f. Veieu la següent taula d'equivalència entre unitats de base 16 (hexadecimal) i base 10 (decimal):

| Hexadecimal | Decimal |
|---|---|
| 0 | 0 |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| a | 10 |
| b | 11 |
| c | 12 |
| d | 13 |
| e | 14 |
| f | 15 |

El valor hexadecimal d'un color es representa amb el símbol # seguit de tres o sis valors hexadecimals.

Aquests sis valors hexadecimals representen, en grups de dos, el valor d'un canal de color específic: *vermell*, *verd* i *blau*. Per exemple, el següent valor hexadecimal #a3ff01 correspon a un color en el que el canal vermell és igual a #a3 (163 en decimal), el canal verd és igual a #ff (255 en decimal) i, finalment, el canal blau és igual a #01 (1 en decimal).

Segons la combinació de valors triada per a cada canal de color, s'obtindrà un color específic dels 16 milions de colors possibles (*255*255*255 = 16.581.375 possibilitats de colors*).


Tres valors hexadecimals és una representació reduïda dels sis valors hexadecimals, quan els valors de cada un dels tres canals estan formats per dos dígits o lletres iguals. Per exemple, el valor hexadecimal #afa equival a #aaffaa, o #8a0 és igual a #88aa00

En el següent exemple, s'han utilitzat les unitats hexadecimals per a especificar el color dels paràgrafs i el color de fons per mitjà d'una regla CSS:

```
p {
color: #aaff00;
background-color: #f00;
}
```

### 7.3. RGB paramètric

La tercera forma d'aplicar color es basa en l'anterior: barrejant diferents quantitats de colors primaris. La diferència està en la sintaxi, ja que en aquest cas no s'usa notació hexadecimal; el que s'usa són coordenades paramètriques.

La sintaxis del valor de la propietat comença per unes sigles, les inicials de cada color que es representa: Red, Green i Blue. Entre parèntesi posem els paràmetres de cada color separats per comes; el rang va de 0 a 255.

```
p {
```

```
color: rgb(170, 255, 0);
}
```

### 7.4. RGB percentual

Existeix una variant que consisteix a posar els paràmetres en tant per cent, en comptes d'utilitzar el rang de 0 a 255. Per exemple, podem escriure la següent declaració per a posar un color vermell: RGB (100%,0%,0%).

```
p {
background-color: rgb(100%, 0, 0);
}
```

A la següent taula podem veure el codi equivalent en el tres sistemes per a diferents colors:

| RELACIÓ DE VALORS PER A CADA COLOR | | | | |
|---|---|---|---|---|
| **color** | **nom** | **hexadecimal** | **paramètric** | **percentual** |
| **vermell** | red | #FF0000 | RGB (255,0,0) | RGB (100%,0%,0%) |
| **llima** | lime | #00FF00 | RGB (0,255,0) | RGB (0%,100%,0%) |
| **blau** | blue | #0000FF | RGB (0,0,255) | RGB (0%,0%,100%) |
| **negre** | black | #000000 | RGB (0,0,0) | RGB (0%,0%,0%) |
| **blanc** | white | #FFFFFF | RGB (255,255,255) | RGB (100%,100%,100%) |
| **groc** | yellow | #FFFF00 | RGB (255,255,0) | RGB (100%,100%,0%) |
| **cian** | aqua | #00FFFF | RGB (0,255,255) | RGB (0%,100%,100%) |
| **rosa** | pink | #FFC0CB | RGB (255,192,203) | RGB (100%,75%,80%) |
| **magenta** | magenta | #FF00FF | RGB (255,0,255) | RGB (100%,0%,100%) |

## 8. Model de caixa

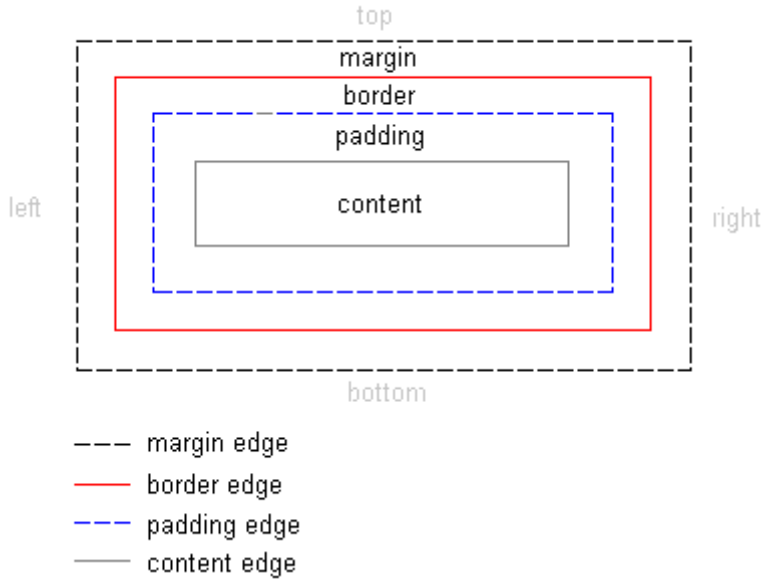
El model de caixes (o capes), en anglès *box model*, és segurament la característica més important del llenguatge de fulls d'estils CSS, ja que condiciona el disseny de totes les pàgines web.

El model de caixes és el comportament de CSS que fa que tots els elements de les pàgines es representin mitjançant caixes rectangulars. Cada vegada que s'insereix una etiqueta HTML es crea una caixa rectangular de forma automàtica que tanca els continguts d'aquest element. Les caixes de les pàgines no són visibles a simple vista perquè inicialment no mostren cap color de fons ni cap vora. Els navegadors creen i col·loquen les caixes de manera automàtica, però CSS permet modificar totes les seves característiques.

### Construcció de les caixes

El model de caixa defineix les caixes que es generen a partir dels elements HTML. També conté opcions detallades pel que fa a l'ajust dels marges, vores, emplenament (padding) i contingut de cada element.

La construcció del model de caixa segueix el següent esquema:




Les parts que componen cada caixa i el seu ordre de visualització des del punt de vista de l'usuari són les següents:

**Content** (contingut) — es tracta del contingut html de l'element (les paraules d'un paràgraf, una imatge, el text d'una llista d'elements, etc.)

| | |
|---|---|
| **Padding** (emplenament) | espai lliure opcional existent entre el contingut i la vora. |
| **Border** (vora) | línia que tanca completament el contingut i el seu emplenament |
| **Margin** (marge) | separació opcional existent entre la caixa i la resta de caixes adjacents |

L'emplenament i el marge són transparents, de manera que a l'espai ocupat per l'emplenament es mostra el color o imatge de fons (si estan definits) i a l'espai ocupat pel marge es mostra el color o imatge de fons del seu element pare (si estan definits).

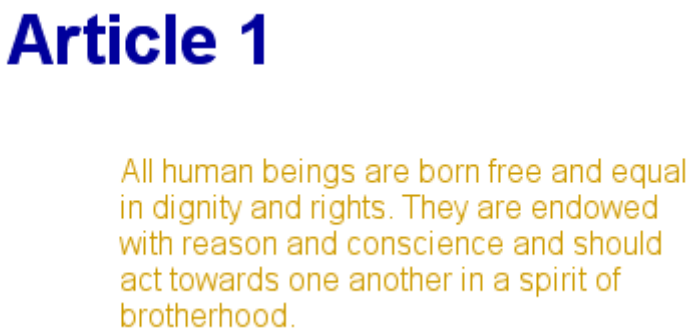
Si cap element pare té definit un color o imatge de fons, es mostra el color o imatge de fons de la pròpia pàgina (si estan definits).

Si una caixa defineix tant un color com una imatge de fons, la imatge té més prioritat i és la que es visualitza. No obstant això, si la imatge de fons no cobreix totalment la caixa de l'element o si la imatge té zones transparents, també es visualitza el color de fons. Combinant imatges transparents i colors de fons es poden aconseguir efectes gràfics molt interessants.

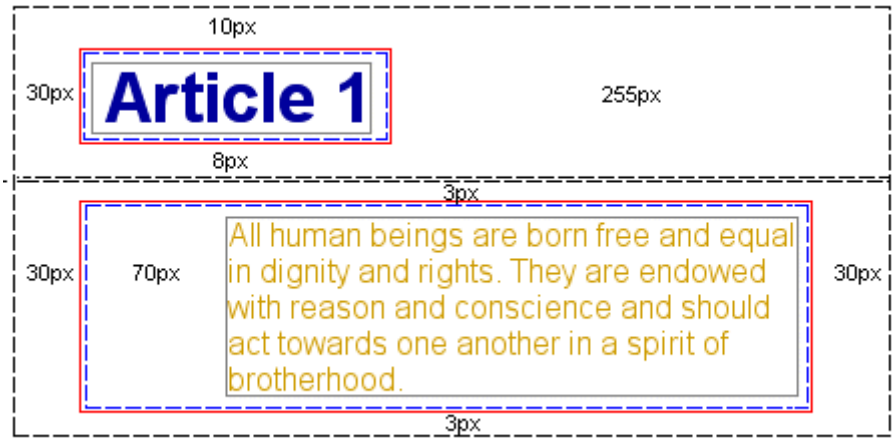
Exemple:

```
<h1>Article 1:</h1>
<p>All human beings are born free and equal in dignity and
rights.They are endowed with reason and conscience and should
act towards one another in a spirit of brotherhood.</p>
```

Si afegim color i característiques del tipus de lletra, l'exemple es podria veure així:



Aquest conté dos elements : <h1> i <p>. El model de caixa per als dos elements es podria il·lustrar de la següent manera:




La imatge mostra com cada un dels elements HTML està envoltat per caixes que es poden ajustar mitjançant CSS.

## 9. Posicionament

CSS permet situar els elements definits als documents XML, HTML, XHTML, ... segons el disseny desitjat mitjançat l'ús de la propietat **position**.

L'aplicació d'aquesta propietat i d'altres tècniques que permeten un control total de l'organització dels continguts de text, imatges, vídeo, etc, als espais de la interfície, rep el nom de maquetació.

Els elements es poden posicionar de cinc formes diferents; normal o estàtic, relatiu, absolut, fix i flotant.

### 9.1. Classificació d'elements: elements en línia i elements de bloc

Segons l'estàndard HTML els elements es classifiquen fonamentalment en dos grups: elements en línia (*inline elements)* i elements de bloc (*block elements)*.

Els elements ***inline*** ...

- resten en línia (horitzontalment) amb altres elements
- defineixen les seves dimensions en funció del seu contingut
- només poden contenir elements *inline*
- no és possible aplicar-lis unes dimensions fixes mitjançant CSS

Dins aquesta tipologia en trobem el següents:

a, abbr, acronym, b, basefont, bdo, big, br, cite, code, dfn, em, font, i, img, input, kbd, label, q, s, samp, select, small, span, strike, strong, sub, sup, textarea, tt, u, var.

Els elements ***block*** ...

- formen blocs i es situen verticalment amb un salt de línia
- pren l'amplada màxima que poden prendre dins el seu element contenidor
- l'alçària canvia en funció de contingut
- poden contenir tant elements *inline* com a *block*
- se'ls hi pot aplicar amplària i alçària fixe amb CSS

A aquesta tipologia pertanyen els següents elements:

address, blockquote, center, dir, div, fieldset, form, h1, h2, h3, h4, h5, h6, hr, isindex, li, menu, noframes, ol, p, pre, table, ul

Amb CSS podem canviar el tipus d'element i per extensió el seu comportament amb la propietat *display* i els valors *inline* i *block*.

Exemple:

```
li {
display: inline;
```

```
}
```

### 9.2. Posicionament estàtic (o normal)

És el que es fa servir per defecte pels navegador per mostrar els elements a una pàgina, ***position: static*** . Un element posicionat d'aquesta forma es diu que està no posicionat. Amb aquest posicionament només es considera:

- si l'element és *inline* o *block*
- les seves propietats *widht* i *height*
- *el seu contingut*

Amb aquest valor, l'element respectarà el flux normal de la pàgina, és a dir, es posicionarà en el lloc que li correspon i no tindrà en compte els valors per a les propietats ***top***, ***left***, ***right*** *i* ***bottom***.

Quan un element està creat dins d'un altre, es fa referència a l'element pare com a element contenidor i determina la posició i la grandària dels elements interiors. En canvi, si un element no està contingut per cap altre, el seu contenidor és l'element *body*.

```
<body>
      <div>
            <div>contingut</div>
      </div>
</body>
```

L'amplària dels elements *block* està limitada normalment a la de l'element pare (o contenidor), tot i que en algun cas el seu contingut pot superar l'espai disponible.

### 9.3. Posicionament relatiu

Amb ***position: relative*** l'element respectarà el flux normal de la pàgina però a diferència de *position*: *static* tindrà en compte els valors per a les propietats ***top***, ***left***, ***right*** *i* ***bottom***.

### 9.4. Posicionament absolut

Aquest valor també acceptarà els valors *top*, *left*, *right* i *bottom*. L'element amb ***position: absolute*** no estarà dins del flux normal de la pàgina i prendrà com a referència la finestra del navegador o l'element posicionat (que tingui qualsevol valor de *position* excepte *static*) més proper si és pare de l'element que volem posicionar.

### 9.5. Posicionament fix

Els elements als quals se'ls posiciona amb ***position: fixed*** també estan fora del flux normal de la pàgina. A diferència dels posicionats amb *position*: *absolute*, els elements amb *position: fixed* prenen com a referència la finestra del navegador i no respecten el

tenir un contenidor pare que estigui posicionat.

En fer *scroll* vertical a la pàgina, l'element que estigui posicionat com *position: fixed* seguirà en la mateixa posició respecte a la finestra del navegador encara que el *scroll* hagi desplaçat la pàgina cap avall.

### 9.6. Posicionament sticky

La posició *sticky* (***position: sticky***), es comporta de forma semblant a un posicionament relatiu però a diferència d'aquest quan arriba a un punt determinat es fixa a la finestra.

Un exemple d'ús habitual per a aquesta propietat serien els elements de menú o el *footers* que han de romandre en una posició relativa a altres elements independentment de la quantitat de contingut del document.

### 9.7. Posicionament flotant

La propietat float permet posicionar els elements de forma flotant. Es pot aplicar a qualsevol de les posicions estudiades excepte *absolute*.

La principal característica del posicionament amb *float* es que la resta d'elements de la pàgina s'adapten per fluir al voltant de la caixa. Aquesta propietat es pot aplicar al següents elements: contenidors (div), img (imatges), p (paràgrafs) i llistes (ul / ol).

El quatre possibles valors son; *left, right, none, inherit*

### 9.8. Superposició d'elements

Alguns elements poden anar cavalcats respecte d'altres. La propietat *z-index* s'empra per ordenar aquests elements de tal manera que es visualitzin segons ens interessi. Aquesta propietat permet controlar els elements, portant-los cap a vall o cap a munt, com si la pàgina tingués tres dimensions.

## 10. Selectors avançats

Utilitzant només els selectors bàsics dels apartats anteriors, és possible dissenyar pràcticament qualsevol pàgina web. Tot i així, CSS defineix altres selectors més sofisticats que permeten simplificar els fulls d'estils.

Cal tenir en compta que tots els navegadors no són compatibles amb aquests tipus de selectors. Al següent enllaç podem comprovar la compatibilitat dels selectors amb el diversos navegadors.

> **Can I use ___ ?**
> https://caniuse.com/

### 10.1. Selector de fills

Es fa servir per seleccionar un element que és fill directe d'un altre element i s'indica mitjançant el "signe de més gran que" (**>**):

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

A l'exemple anterior, el selector **p > span** s'interpreta com "qualsevol element **** que sigui fill directe d'un element **<p>**", per la qual cosa el primer element **** compleix la condició del selector. Un segon element **** no la compliria perquè seria descendent però no fill directe d'un element **<p>**.

El següent exemple mostra les diferències entre el selector descendent i el selector de fills:

```
p a {color:red;}
p > a {color:red;}

<p><a href="#">Enllaç</a></p>
<p><a href="#"><span>Text</span></a></p>
```

El primer selector és de tipus descendent i per tant s'aplica a tots els elements **<a>** que es troben dins d'elements **<p>**.

D'altra banda, el selector de fills obliga al fet que l'element **<a>** sigui fill directe d'un element **<p>**. Per tant, els estils del selector **p > a** no s'apliquen a l 'element **span** de l'exemple anterior.

### 10.2. Selector adjacent

El selector adjacent s'empra per seleccionar elements que en el codi HTML de la pàgina es troben just a continuació d'altres elements. La seva sintaxi fa servir el signe **+** per separar els dos elements:

```
element1 + element2 { ... }
```

Si es considera el següent codi HTML:

```
<body>
      <h1>Títol1</h1>
      <h2>Subtítol1</h2>
      ...
      <h2>Subtítol2</h2>
      ...
</body>
```

La pàgina anterior disposa de dos elements **<h2>**, però només un d'ells es troba immediatament després de l'element **<h1>**. Si es vol aplicar diferents colors en funció d'aquesta circumstància, el selector adjacent és el més adequat:

```
h2 { color: green; }
h1 + h2 { color: red; }
```

Les regles CSS anteriors fan que tots els **h2** de la pàgina es vegin de color verd, excepte aquells que es troben immediatament després de qualsevol element **<h1>** que es mostren de color vermell.

Tècnicament, els elements que formen el selector adjacent han de complir les següents condicions:

- **element1** i **element2** han de ser elements germans, això vol dir que el seu element pare ha de ser el mateix
- **element2** ha d'aparèixer immediatament després d'**element1** en el codi HTML de la pàgina

### 10.3. Selector d'atributs

Un tercer tipus de selectors avançats ho formen els selectors d'atributs, que permeten seleccionar elements HTML en funció dels seus atributs i/o valors d'aquests atributs.

Els quatre tipus de selectors d'atributs són:

- **[nom_atribut]**, selecciona els elements que tenen establert l'atribut anomenat **nom_atribut**, independentment del seu **valor**.

  ```
  a[class] {color:green;}
  ```

  A l'exemple es mostren de color vermell tots els enllaços que tinguin un atribut "class", independentment del seu valor

- **[nom_atribut=valor]**, selecciona els elements que tenen establert un atribut anomenat **nom_atribut** amb un valor igual a **valor**.

  ```
  a[class="extern"] {color:red;}
  ```

  A l'exemple es mostren de color vermell tots els enllaços que tinguin un atribut "class" amb el valor "extern"

  ```
  a[href="http://http://www.w3.org/"] {color:red;}
  ```

  Es mostren de color vermell tots els enllaços que apuntin al lloc "http://www.w3c.org"

- **[nom_atribut~=valor]**, selecciona els elements que tenen establert un atribut anomenat **nom_atribut** i almenys un dels valors de l'atribut és **valor**.

  ```
  a[class~="extern"] {color:red;}
  ```

  A l'exemple es mostren de color vermell tots els enllaços que tinguin un atribut "class" en el qual almenys un dels seus valors sigui "extern"

- **[nom_atribut|=valor]**, selecciona els elements que tenen establert un atribut anomenat **nom_atribu**t i el valor del qual és una sèrie de paraules separades amb guions, però que comença amb **valor**.

  Aquest tipus de selector només és útil per als atributs de tipus **lang** que indiquen l'idioma del contingut de l'element.

  ```
  *[lang=en] { ... }
  ```

  Selecciona tots els elements de la pàgina l'atribut de la qual"lang" sigui igual a "en", és a dir, tots els elements en anglès.

  ```
  *[lang|="es"] {color:red;}
  ```

Selecciona tots els elements de la pàgina l'atribut de la qual "lang" comenci per "es", és a dir, "es", "es-ES", "es-AR", etc.

### 10.4. Pseudo-classes

Una pseudo-classe, és una paraula clau que podem afegir als selectors CSS emprant dos punts ( : ) amb l'objectiu de definir un estat especial de l'element HTML al que pertany.

La sintaxi emprada per aquesta eina és:

```
selector:pseudo-classe {propietat: valor;}
```

Els navegadors antics no permeten l'aplicació de les **pseudo-classes** a elements que no siguin enllaços. Es recomana revisar la compatibilitat amb les versions dels navegadors abans de fer ús de les pseudo-classes.

Entre d'altres, existeixen les següents pseudo-classes:

- **Pseudo-classe :first-child.** Selecciona el primer fill d'un element pare.

  Exemple:

```
p:first-child {
      font-size: 16px;
      color: #FFF000;
}

<div>
  <p>XML</p>
  <p>HTML</p>
  <p>CSS</p>
</div>
```

  Al exemple, l'element seleccionat seria XML . Es a dir, del elements fills de  el format només s'aplicaria a aquest.

- **Pseudo-classe :last-child.** Funciona de forma anàloga a **:first-child**, seleccionant en aquest cas l'ultim fill de l'element pare.

  Exemple:

```
p:last-child {
      font-size: 16px;
      color: #FFF000;
}

<div>
  <p>XML</p>
  <p>HTML</p>
  <p>CSS</p>
</div>
```

  Al exemple l'element seleccionat seria CSS .

- **Pseudo-classe :lonly-child.** Selecciona element que són fills únics, es a dir que no tenen cap germà.

  Exemple:

```
p:only-child {
      font-size: 16px;
      color: #FFF000;
}

<div>
  <p>XML</p>
  <p>HTML</p>
  <p>CSS</p>
</div>

<div>
  <p>Java</p>
</div>
```

  Al exemple anterior l'únic element seleccionat seria Java , atès que XML , HTML i CSS són germans fills de  i per tant no són fills únics.

- **Pseudo-classe :nth-child().** La pseudo-classe **:nth-child(an+b)** permet la selecció d'un element o d'una sèrie d'elements fills que es troben dins d'un element pare.

  Exemple:

```
p:nth-child(0n+1){
      font-size: 16px;
      color: #FFF000;
}

<div>
  <p>XML</p>
  <p>HTML</p>
  <p>CSS</p>
</div>
```

  Al exemple l'element seleccionat seria HTML .

  En el cas que volguéssim seleccionar els elements parells o senars d'una sèrie d'elements farem servir els valors **odd** i **even** respectivament, **:nth-child(odd)** i **:nth-child(even)**.

  Exemple:

```
p:nth-child(even){
      font-size: 16px;
      color: #FFF000;
}
```

```
<div>
  <p>XML</p>
  <p>HTML</p>
  <p>CSS</p>
</div>
```

A l'exemple anterior els elements seleccionats serien XML i CSS .

A la web NTH-TEST es pot observar els resultats de l'ús d'aquesta pseudo-classe de forma visual.

- **Pseudo-classe :link.** La pseudo-classe **:link** s'aplica a tots els enllaços que encara no s'han visitat.

  Per canviar aquest format (subratllat de color blau) s'ha de canviar amb la pseudo-classe **:link**.

  Exemple:

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

- **Pseudo-classe :visited.** La pseudo-classe **:visited** s'aplica a tots els enllaços que s'han visitat com a mínim un cop.

  Per canviar aquest format (subratllat de color porpre) s'ha de canviar amb la pseudo- classe **:visited**.

  Les pseudo-classes **:link** i **:visited** són excloents entre si, és a dir, un mateix enllaç no pot experimentar tots dos estats de forma simultània.

  Exemple:

```
a:visited {
      color: #000;
}
```

- **Pseudo-classe :hover.** Aquesta pseudo-classe s'activa quan el cursor passa per sobre d'un element.

  Exemple:

```
h1:hover {
      background-color: #808b96;
}
```

  En aquest cas l'efecte *hover* aplicat canviaria el fons del titular al color `#808b96`.

- **Pseudo-classe :active.** S'activa quan l'usuari activa un element, com ara prémer un boto de la interfície.

  L'efecte és gairebé imperceptible, ja que només dura mentre es prem el botó del ratolí a sobre de l'enllaç.

  Exemple:

```
a:active {
      color: #000;
}
```

- **Pseudo-classe :focus.** Aquesta pseudo-classe s'activa un element que té el focus del navegador.
  Quan estem emplenant un formulari, normalment premem la tecla TAB per a canviar al següent camp i SHIFT+TAB per tornar enrere. Al prémer la tecla TAB i situar-nos al camp es diu que aquest **té el focus**, mentre que si passem a un altre camp es diu que **perd el focus**.

  Es fa servir principalment amb formularis però també amb enllaços.

  Exemple:

```
input:focus {
      background-color: #eaecee;
}
```

**Llista de les pseudo-classes estàndard**
(en negreta les explicades)

| | | |
|---|---|---|
| **:active** | :indeterminate | **:only-child** |
| :checked | :in-range | :only-of-type |
| :default | :invalid | :optional |
| :dir() | :lang() | :out-of-range |
| :disabled | **:last-child** | :read-only |
| :empty | :last-of-type | :read-write |
| :enabled | :left | :required |
| :first | **:link** | :right |
| **:first-child** | :not() | :root |
| :first-of-type | **:nth-child()** | :scope |
| :fullscreen | :nth-last-child() | :target |
| **:focus** | :nth-last-of-type() | :valid |
| **:hover** | :nth-of-type() | **:visited** |

### 10.5. Pseudo-elements

De igual forma que les pseudo-classes, els pseudo-elements s'afegeixen als selectors, però a diferència de les primeres no descriuen un estat de l'element sinó que permeten donar estil a una part d'aquest.

La sintaxi emprada és la següent:

```
selector::pseudo-element {propietat: valor;}
```

Només es pot emprar un pseudo-element per selector, i ha d'aparèixer després d'aquest. Amb els pseudo-elements solen fer-se servir els dobles punts dobles :: per a distingir-los de les pseudo-classes (es va introduir amb CSS3), tot i que es poden fer servir també els punts dobles menys amb els pseudo-element **::selection**.

```
p::first-letter {
      color: red;
}
```

Els navegadors antics no permeten l'aplicació d'alguns pseudo-elements. Es recomana revisar la compatibilitat amb les versions dels navegador abans de fer ús d'aquests.

CSS selectors: basic browser support

Entre d'altres, podem fer servir els següents pseudo-elements:

- **Pseudo-element ::before i ::after.** Aquests pseudo-elements es combinen amb la propietat content: per afegir continguts abans o després del contingut propi de l'element. Aquest contingut pot ser text, imatges, ...

  Exemple:

  ```
  h1::before {
        content: "Capítol - ";
  }
  ```

  El resultat seria que abans de l'etiqueta h1 apareixeria sempre contingut definit, en aquest cas, **Capítol 1**.

- **Pseudo-element ::first-letter.** Aquest pseudo-element dona estil a la primera lletra d'un element.

  Exemple:

  ```
  p::first-letter {
        color: #808b96;
  }
  ```

En aquest cas la primera lletra de tots els paràgrafs seria de color `#808b96`.

- **Pseudo-element ::first-line.** Aquest pseudo-element dona estil a la primera línia d'un element.

  Exemple:

  ```
  p::first-line {
      font-weight: bold;
  }
  ```

  En aquest cas la primera línia de tots els paràgrafs estarien representada en negreta.

**Llista de les pseudo-elements**
(en negreta les explicades, en vermell les que encara són en fase experimental)

| | | |
|---|---|---|
| **::after** | **::first-line** | ::selection |
| ::backdrop | ::grammar-error | ::spelling-error |
| **::before** | ::marker | |
| **::first-letter** | ::placeholer | |

## 11. Novetats a CSS3

CSS3 és modular atès que l'especificació d'estils va ser créixer tant, que es va haver de desenvolupar de forma modular. De fet és una especificació que es desenvolupa des del 1999.

Cada mòdul de CCS3 té el seu grau d'evolució, així que l'estandardització dels mòduls és propi. Podem tenir un mòdul que sigui ja estàndard i un altre mòdul que estigui encara en un procés d'estandardització. De fet s'estima que no hi haurà una versió de CSS4, sinó que hi haurà mòduls en la seva versió 4 i mòduls que encara no hauran progressat.

Entre les coses que podem fer amb CSS3 destaquem algunes:

- Permet fer molt fàcilment coses com les **vores arrodonides**, que amb versions anteriors de CSS3 eren bastant difícils d'aconseguir.
- Permet donar **gradients de colors**, és a dir, si volem donar a una etiqueta o a una part de la pàgina web un color de fons, ja no ens veiem limitats a donar-li un únic color, sinó que podem establir un gradient, una transició d'un color a un altre.
- Es poden fer **transformacions** en moltes de les propietats dels elements de la pàgina web i fins i tot animacions utilitzant únicament CSS3.
- Aplicar **ombres** a elements i text.
- Organitzar el text en diverses **columnes**.
- **Deformar** elements HTML sense haver de fer servir imatges.
- Permet **maquetar de manera molt més fàcil** utilitzant contenidors Flex i Grid.
- Permet utilitzar les **media-queries**, que permeten triar un full d'estils depenent de les propietats de la pantalla que tinguem, perquè la pàgina web canviï la seva presentació si es mostri en una pantalla gran, en un mòbil, en una tauleta i fins i tot en un televisor.

CSS3 **encara no és un estàndard**, sinó que ha evolucionat i que cada mòdul té el seu propi grau d'evolució.

Si volem veure fins a quin punt un navegador suporta CSS3 podem visitar la web: CSS3 Test, on s'indica el percentatge de suport que té el navegador per CSS3 en general i per a cada mòdul concret.

### 11.1 Vores arrodonides.

La propietat **border-radius** ens permet aplicar cantonades arrodonides. Aquest recurs de disseny abans s'havia de fer amb JavaScript o imatges.

Exemple:

```
.border-radius {
        border-radius: 20px;
        }
```

A l'exemple s'han aplicat arrodoniment dels vèrtex amb un radi de 20 píxels.

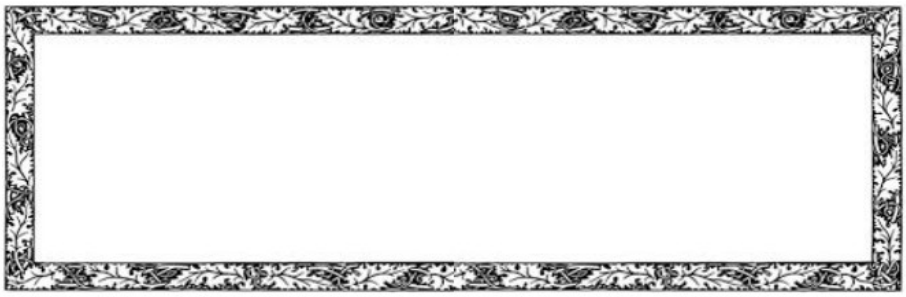
http://www.w3schools.com/cssref/css3_pr_border-radius.asp

### 11.2 Vores amb imatges

La propietat border-image permet l'ús d'imatges per aplicar-les a les vores.

Exemple:

```
.border-image {
                border: 30px solid transparent;
                border-image: url(border.jpeg) 25 25 round;
        }
```



https://www.w3schools.com/css/css3_border_images.asp

## 11.3 Degradats de colors

Permet el canvi progressiu d'un color a un altre d'un element (per exemple d'un contenidor) i fins i tot es poden emprar diferents colors. A més els degradats poden ser lineals, amb diferent direccions o radials, fer servir transparències, ...

Exemple:

```
.deg {
        background-image: linear-gradient( 45deg, yellow, black);
        }
```

A l'exemple apliquem un degradat lineal de color, de groc a negre, amb un angle de 45º

http://www.w3schools.com/css/css3_gradients.asp

## 11.4 Ombres

La propietat **box-shadow** permet aplicar ombres a les caixes definides amb CSS.

Exemple:

```
.boxshadow {
        box-shadow: 10px 10px 5px 5px rgba(100, 100, 100, 0.5);
        }
```

A l'exemple s'aplica ombra a la caixa amb els següents paràmetres; 10 píxels en els eixos horitzontal i vertical amb una degradació de 5px, el color de l'ombra seria el negre amb una transparència de 0.5.

https://www.w3schools.com/cssref/css3_pr_box-shadow.asp

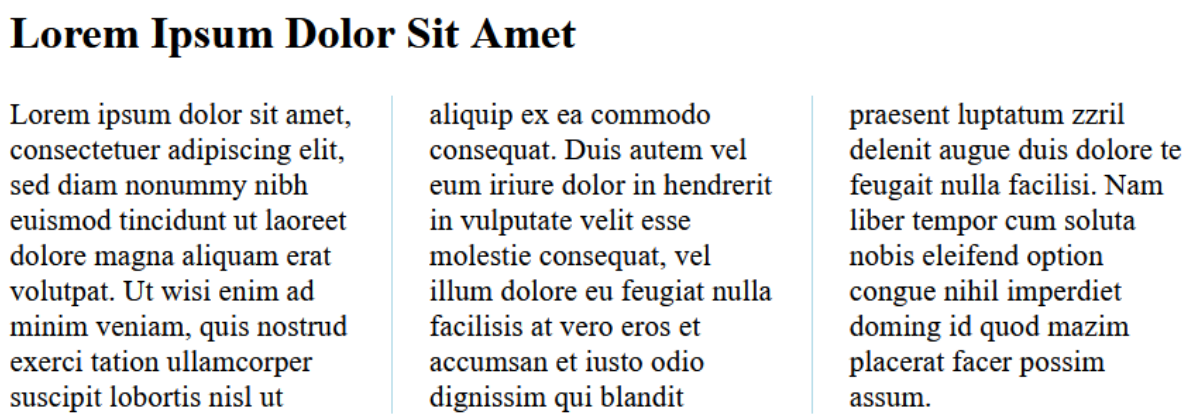
## 11.5 Columnes múltiples

Amb CSS3 podem repartir horitzontalment el text en diferents columnes. Es pot definir el nombre de columnes, la distància entre columnes, l'ús de pautes verticals, etc.

Exemple:

```
.columnes {
        column-count: 3;
        column-gap: 40px;
        column-rule: 1px solid lightblue;
}
```

A l'exemple s'ha distribuït el text en tres columnes, amb una separació de 40 píxels entre columnes i s'ha aplicat una línia vertical de separació d'un píxel de gruix de color *lightblue*.



**Lorem Ipsum Dolor Sit Amet**

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

https://www.w3schools.com/css/css3_multiple_columns.asp

## 11.6 Transformacions (2D i 3D)

Amb CSS3 s'han afegit eines per a la transformació 2D com ara; moviments, rotacions, escalats, ... i transformacions 3D com ara rotacions en diferents eixos.

Exemple:

```
.rotate {
        transform: rotate(30deg);
}
```

A l'exemple es mostra una caixa amb una rotació de 30º.


https://www.w3schools.com/css/css3_2dtransforms.asp
https://www.w3schools.com/css/css3_3dtransforms.asp

### 11.7 Transicions

Abans de CSS3, amb pseudo-classes (per exemple amb *:hover),* ja es podien obtenir modificacions d'un element de la interfície des d'un punt inicial a un final de forma sobtada. Amb les transicions es pot passar d'unes característiques a unes altres de forma suau.

Exemple:

```
.transicio {
        background-color: red;
        transition-property: background-color;
        transition-duration: 1s;
        transition-timing-function: linear;
        transition-delay: 1.5s;
}
```

A l'exemple l'element experimentarà un canvi del color de fons al color vermell en un temps d'un segon de durada, de forma lineal, amb un retard de 1,5 segons des del moment que s'aplica el canvi.

https://www.w3schools.com/css/css3_transitions.asp
https://lenguajecss.com/css/animaciones/transiciones/

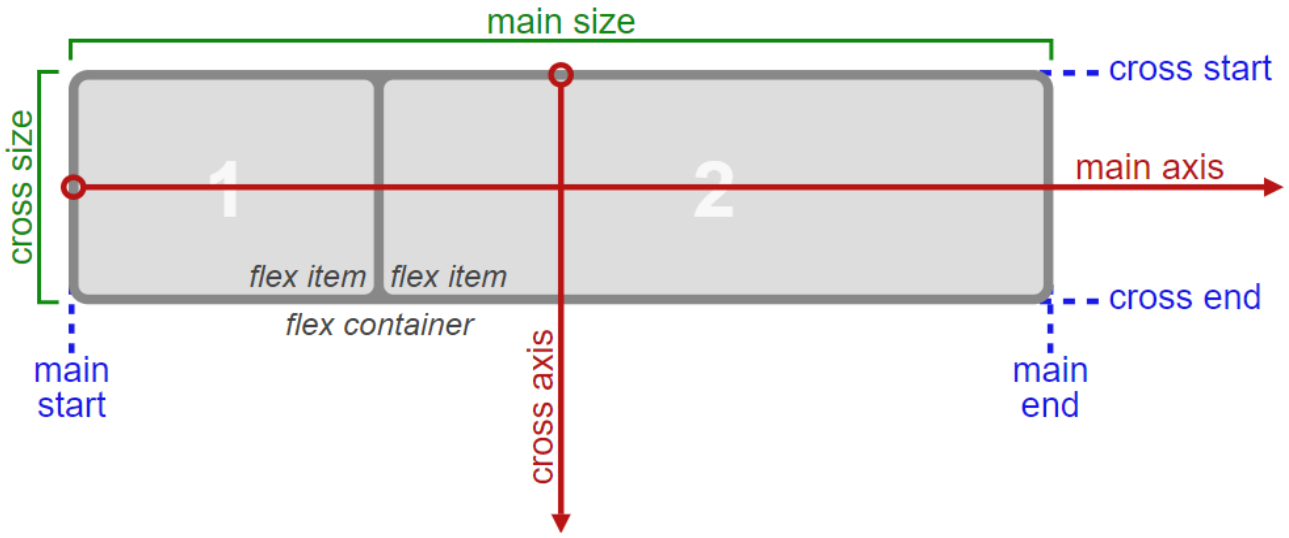
## 11.8 Flexbox

Flexbox i Grid són dos mòduls de disseny de CSS que permeten la creació de dissenys complexos per a interfícies que abans només eren possibles mitjançant javascript o emprant tècniques molt avançades.

És un recurs de CSS que dona suport a la distribució dels espais ocupats a la interfície per millorar l'alineació dels ítems (imatges, icones, agrupacions de text, etc).

Es va introduir l'any 2009 per millorar el desenvolupament de pàgines *responsive*.

Aquesta eina permet el control unidireccional del *layout*, això vol dir que regula els elements en una sola direcció cada cop, a diferència d'altres eines com ara Grid Layout de CSS, que controla files i columnes a l'hora.




Enllaços d'interès:

Conceptos Básicos de flexbox
https://developer.mozilla.org/es/docs/Web/CSS/CSS_Flexible_Box_Layout/Conceptos_Basicos_de_Flexbox

Flexbox a base de ejemplos
https://lemoncode.net/lemoncode-blog/2016/12/28/flexbox-a-base-de-ejemplos

Flexbox CSS
https://lenguajecss.com/p/css/propiedades/flexbox#propiedades-de-alineacin-de-tems

Maquetación Flex
http://w3.unpocodetodo.info/css3/flex-maquetacion.php

Flexbox playground
https://codepen.io/enxaneta/full/adLPwv/

CSS Flexbox Examples
https://www.quackit.com/css/flexbox/examples/

Flexbox CSS: Guia Completo, Elementos y Ejemplos
https://www.aluracursos.com/blog/flexbox-css-guia-completo-elementos-y-ejemplos

### 11.9 Grid Layout

CSS Grid Layout és un sistema de graella pensat, sobretot, per a treballar en dues dimensions. Aquesta tecnologia té una bona compatibilitat amb la majoria de navegadors, no cal afegir cap element per que el navegador ho pugui interpretar.

Es va desenvolupar amb l'objectiu de millorar Flexbox i estendre el seu ús més enllà de la creació de barres horitzontals i verticals d'elements i arribar al seu ús per a la maquetació completa de pàgines amb disseny *responsive*.

| Propietat | Descripció |
|---|---|
| column-gap | Indica l'amplada de les columnes |
| gap | Indica l'espai entre files i columnes |
| grid | Forma resumida de les propietats *grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns*, *grid-auto-flow* |
| grid-area | Forma resumida de les propietats *grid-row-start*, *grid-column-start*, *grid-row-end*, i *grid-column-end* |
| grid-auto-columns | Indica que la amplada és automàtica per les columnes |
| grid-auto-flow | Indica com es s'integren a la graella els ítems quan no s'integren amb cap propietat específica |
| grid-auto-rows | Indica que l'alçària és automàtica per les files |
| grid-column | Forma resumida de les propietats *grid-column-start* , *grid-column-end* |
| grid-column-end | Indica a quina columna acaba l'ítem de la graella |
| grid-column-gap | Indica l'espai entre columnes |
| grid-column-start | Indica a quina columna comença l'ítem de la graella |
| grid-gap | Forma resumida de les propietats *grid-row-gap* i *grid-column-gap* |
| grid-row | Forma resumida de les propietats *grid-row-start* i the *grid-row-end* |
| grid-row-end | Indica a quina fila acaba l'ítem de la graella |
| grid-row-gap | Indica l'espai entre files |
| grid-row-start | Indica a quina fila comença l'ítem de la graella |
| grid-template | Forma resumida de les propietats *grid-template-rows*, *grid-template-columns* i *grid-areas* properties |
| grid-template-areas | Indica el nom de cada una de les àrees de la graella |
| grid-template-columns | Indica l'amplada de les columnes i el nombre de columnes de la graella |
| grid-template-rows | Indica la mida de les files dins d'una graella |
| row-gap | Indica l'alçària de les files |

Grid CSS (Cuadrículas)
https://lenguajecss.com/css/maquetacion-y-colocacion/grid-css/

A complete guide to Grid
https://css-tricks.com/snippets/css/complete-guide-grid/

Grid by example
https://gridbyexample.com/examples/page-layout/#layout1

Todo lo que necesitas saber de CSS Grid
https://platzi.com/tutoriales/1229-css-grid-layout-2017/6653-todo-lo-que-necesitas-saber-de-css-grid/

**Quina tecnologia hem de fer servir?**

Cap de les dues opcions és millor que l'altre, es podria dir que:

- CSS Grid es millor per a la construcció general d'una pàgina, atès que és més fàcil d'ús per a la maquetació creant dissenys asimètrics.
- Flexbox és millor alineant els continguts i funciona millor en una sola dimensió

Relationship of grid layout to other layout methods
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Grid_Layout/Relationship_of_Grid_Layout#grid_and_flexbox

## 12. Regles d'arrova (*at-rules*)

Són declaracions CSS que agrupen un grup de regles, tot i que no estan directament relacionades a elements HTML, PHP, ... Regulen la forma en la que s'apliquen els estils ampliant les possibilitats de CSS. Cada regle té la seva pròpia sintaxi.

### 12.1. Regles generals

S'apliquen a tot el CSS

- @charset — Defineix el conjunt de caràcters emprats amb CSS.
- @import — Permet importar regles d'un altre full.
- @namespace — Estableix l'espai de noms XML emprats en un full d'estils.

### 12.2. Regles imbricades

Poden aparèixer com a declaració de estil o com a condicionals.

- @media — Estableix regles condicionals en funció de les característiques del dispositiu (mida de pantalla, tipus d'impressora, etc.)
- @supports — Estableix regles condicionals segons el navegador emprat.
- @document — Restringeix les regles contingudes segons la URL.
- @page — Restringeix les regles contingudes quan es vol imprimir (en impressora) .
- @font-face — Afegeix tipografies externes.
- @keyframes — Controla les fases intermèdies en una seqüencia animada.
- @viewport — Restringeix les regles segons la mida i orientació de la finestra. Orientat a dispositius mòbils.
- @counter-style — Defineix estils de comptadors específics.
- @font-feature-values, *@swash, @ornaments, @annotation, @stylistic, @styleset y @character-variant* — Defineix noms comuns para la propietat *font-variant-alternates*.

Exemples:

1. Definició de regles diferents en funció del document que es carregui amb @document

```
@document url(http://url.com/index.html) {
  body {
    background-image: url("index.png");
  }
```

```
}

@document url-prefix(http://url.com/users) {
  body {
    background-image: url("users.png");
  }
}
```

En aquest exemple definim, dins del mateix full d'estils, una imatge de fons per a la pàgina principal i d'altre per a les que comencen a la pàgina users.

2. Ús de tipografies de Google Fonts

En el següent exemple es carrega la tipografia Montserrat de Google fonts al full d'estils.

```
@font-face {
font-family: "Montserrat";
font-style: normal;
font-weight: 400;
src:url(https://fonts.google.com/specimen/Montserrat?
query=montserrat&sidebar.open=true&selection.family=Montserrat:ital,wght@0,
400;1,100
}
```

A continuació l'apliquem:

```
body {
  font-family: "Montserrat";
  font-weight: 400;
  font-style: normal;
  font-size: 16px;
  }
```

3. Afegir un segon full d'estils CSS per quan el dispositiu estigui apaïsat.

```
@import url('css_2.css') screen and (orientation:landscape);
```