

HTML i CSS

M04 UF1 A1.5

Definir codi CSS

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Cache-Control" content="no-cache, no-store,
must-revalidate" />
    <meta http-equiv="Pragma" content="no-cache" />
    <meta http-equiv="Expires" content="0" />
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta name="viewport" content="initial-scale=1, maximum-scale=1">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link
href="https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,6
00,600i,700,700i,800,800i&display=swap" rel="stylesheet" />
    <link href="estils.css" rel="stylesheet" />
    <title>Exemples CSS</title>
  </head>
  <style>
    body {
      font-family: "Open Sans", sans-serif;
      margin: 0;
      user-select: none;
    }
  </style>
  <body>
  </body>
</html>
```

Hi ha dues maneres per definir codi CSS.

- Carregant-lo d'arxius .css amb 'link' (manera recomanada)

- Posant-lo entre <style></style> (manera **desaconsellada**)

Definicions CSS

Les definicions CSS poden tenir:

- [selector](#): a quin o quins elements afecta aquella definició
- [propietats](#) de la definició: quin aspecte modifiquen
- [colors](#): les propietats de color accepten 'textos', valors amb #hexadecimal, o amb rgba
- [funcions](#): algunes propietats accepten càlculs sobre els valors
- [animacions](#): les propietats numèriques poden definir canvis per fer animacions

```
body {  
  background-color: rgb(100, 100, 100);  
  font-family: "Open Sans", sans-serif;  
  margin: 0;  
  user-select: none;  
}
```

Nota: Si no es defineixen propietats, es posen [les definides per defecte](#)

CSS, selectors per nom d'element

```
<body>  
  <p>poma</p>  
  <p>cotxe</p>  
</body>
```

```
body {  
  background-color: rgb(100, 100, 100);  
  font-family: "Open Sans", sans-serif;  
  margin: 0;  
  padding: 25px;  
  user-select: none;  
}  
p { border: solid 1px red; }
```

poma

cotxe

Quan el selector és un nom d'element, en aquest exemple `<body>` o `<p>`, la definició afecta a tots aquells elements de la pàgina (que no tenen una definició més específica).

A l'exemple es veu com la definició d'estil per a `<p>` afecta a tots els elements `<p>` de la pàgina.

CSS, selectors per id d'element

```
<body>  
  <p>poma</p>  
  <p>cotxe</p>  
  <p id="b00">boli</p>  
</body>
```

```
p {  
  border: solid 1px red;  
}  
#b00 {  
  border: dashed 2px lightgreen;  
}
```

Quan volem definir codi CSS a partir de l'identificador d'element, fem servir **#** seguit de l'identificador.

En aquest exemple **#b00** defineix el CSS de l'element:

```
<p id="b00">boli</p>
```

poma

cotxe

boli

CSS, selectors per **class** d'element

```
<body>
  <p class="rosa">poma</p>
  <p>cotxe</p>
  <p id="b00">boli</p>
</body>
```

```
p {
  border: solid 1px red;
}
#b00 {
  border: dashed 2px lightgreen;
}
.rosa {
  background-color: lightpink;
}
```

poma

cotxe

boli

L'atribut '**class**' ens permet assignar un o diverses definicions CSS a un element. Les definicions es fan amb un punt . seguit del nom de classe

En aquest exemple **.rosa** defineix el CSS de l'element:

```
<p class="rosa">poma</p>
```

CSS, selectors per **class** d'element (múltiple)

```
<body>
  <p class="rosa rounded">poma</p>
  <p>cotxe</p>
  <p id="b00">boli</p>
</body>
```

```
p { border: solid 1px red; }

#b00 { border: dashed 2px lightgreen; }

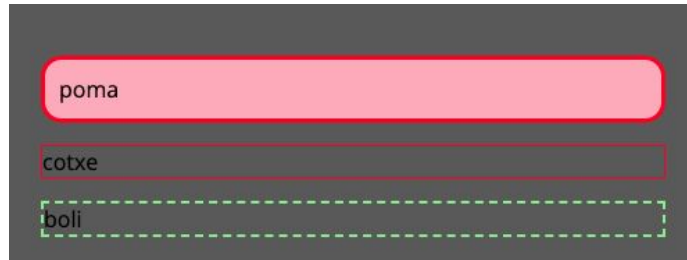
.rosa { background-color: lightpink; }

.rounded {
  border-radius: 15px;
  border: solid 3px red;
  padding: 10px;
}
```

L'atribut '**class**' ens permet assignar múltiples definicions CSS a un element. Les definicions es fan amb un punt . seguit del nom de classe

En aquest exemple **.rosa** i **.rounded** defineixen el CSS de l'element:

```
<p class="rosa rounded">poma</p>
```



CSS, selectors per **atributs** d'element

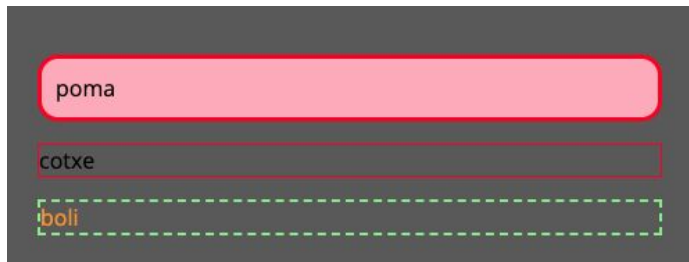
```
<body>  
  <p class="rosa rounded">poma</p>  
  <p>cotxe</p>  
  <p id="b00" data-in="abc">boli</p>  
</body>
```

```
*[data-in="abc"] {  
  color: orange;  
}
```

També es poden fer definicions CSS a partir de valors dels atributs. El format és `*[]` amb l'atribut i el valor.

En aquest exemple `*[data-in="abc"]` defineix el CSS de l'element:

```
<p id="b00" data-in="abc">boli</p>
```



CSS, selectors amb **:hover**

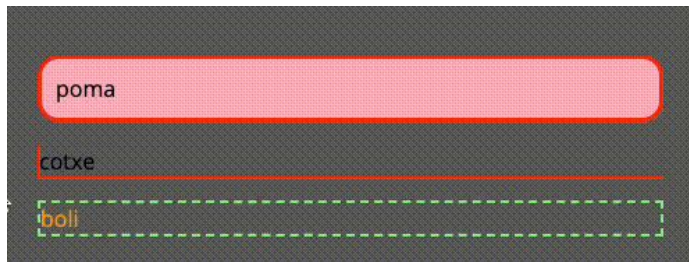
```
<body>
  <p class="rosa rounded">poma</p>
  <p>cotxe</p>
  <p id="b00" data-in="abc">boli</p>
</body>
```

```
.rosa {
  background-color: lightpink;
}
.rosa:hover {
  background-color: yellow;
  cursor: pointer;
}
```

Quan afegim **:hover** a un selector CSS, indiquem que aquelles propietats només s'apliquen si el mouse està per sobre de l'element.

En aquest exemple **.rosa** i **.rounded** defineixen el CSS de l'element i rosa té una definició amb **:hover**

```
<p class="rosa rounded">poma</p>
```



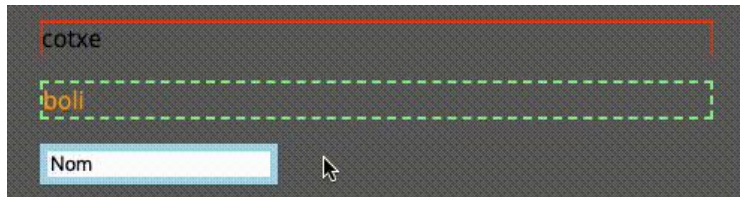
CSS, selectors amb **:focus**

```
<body>
<p class="rosa rounded">poma</p>
<p>cotxe</p>
<p id="b00" data-in="abc">boli</p>
<input class="in00" type="text" value="Nom" />
</body>
```

Quan afegim **:focus** a un selector CSS, indiquem que aquelles propietats només s'apliquen si l'element està actiu.

```
.in00 {
  border: solid 5px lightblue;
  transition: border 0.5s ease;
}
.in00:focus {
  border: solid 10px red;
}
```

En aquest exemple quan l'element `<input>` té el focus se li canvia la propietat 'border'



Selectors, amb comes i espais

```
table, th, td {  
  border: solid 1px black;  
  border-collapse: collapse;  
}  
.t0 { background-color: whitesmoke; }  
.t0 tr:first-child {  
  background-color: black;  
  color: white;  
}  
.t0 tr:nth-child(2) {  
  background-color: lightskyblue;  
}  
.t0 tr:nth-child(5) {  
  background-color: pink;  
}  
.t0 tr:last-child {  
  background-color: lightslategray;  
}
```

Quan es posen selectors separats per comes (exemple: table, th, td), es defineixen propietats CSS per tots ells alhora

Quan es defineix un element, espai un altre element, vol dir que el segon element està contingut dins del primer. No té perquè ser directament

En aquest cas, la taula amb **class="t0"** té fills `<tr>` dins

Selectors, amb :first-child, :nth-child(n) i :last-child

```
<table class="t0">
```

Nom	Cognom
1 Shinchán	Nohara
2 Masao	Sato
3 Nene	Sakurada
4 Himawari	Nohara
5 Misae	Nohara
6 Hiroshi	Nohara

```
table, th, td {  
  border: solid 1px black;  
  border-collapse: collapse;  
}  
.t0 { background-color: whitesmoke; }  
.t0 tr:first-child {  
  background-color: black;  
  color: white;  
}  
.t0 tr:nth-child(2) {  
  background-color: lightskyblue;  
}  
.t0 tr:nth-child(5) {  
  background-color: pink;  
}  
.t0 tr:last-child {  
  background-color: lightslategray;  
}
```

.t0 tr:first-child fa referència al primer element <tr> que depèn de la taula amb class="t0"

Amb **:nth-child(n)**, s'ha de canviar la 'n' pel número amb la posició del fill, començant des de 1

.t0 tr:last-child fa referència a l'últim element <tr> que depèn de la taula amb class="t0"

Selectors, amb `:nth-child(even)` i `:nth-child(odd)`

```
<table class="t1">
```

	Nom	Cognom
1	Shinchan	Nohara
2	Masao	Sato
3	Nene	Sakurada
4	Himawari	Nohara
5	Misae	Nohara
6	Hiroshi	Nohara

```
table, th, td {  
  border: solid 1px black;  
  border-collapse: collapse;  
}  
.t1 tr:nth-child(even) {  
  color: blue;  
}  
.t1 tr:nth-child(odd) {  
  background-color: lightskyblue;  
}
```

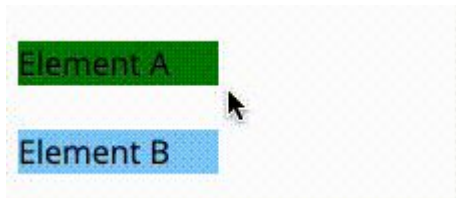
`:nth-child(n)`, a part de números
també accepta

(even) per les posicions parells

(odd) per les posicions imparells

Selectors, amb ~

```
<div id="elmA">Element A</div>  
<br/>  
<div id="elmB">Element B</div>
```



```
#elmA {  
  background-color: green;  
  width: 100px;  
}  
#elmA:hover {  
  cursor: pointer;  
}  
#elmA:hover ~ #elmB {  
  background-color: red;  
  width: 200px;  
}  
#elmB {  
  background-color: lightskyblue;  
  width: 100px;  
}
```

Amb el caràcter tilda ~, es pot fer que l'estat d'un element canviï l'estil CSS d'un altre element

En aquest cas, fer 'hover' a l'element amb id="elmA" canvia les propietats CSS de l'element amb id="elmB"

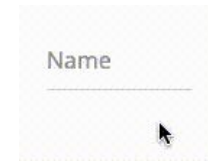
Selectors, amb **:valid** i múltiples modificacions

:valid valida que els camps introduïts en un formulari són correctes o no, i permet mostrar un error en conseqüència (si compleix el pattern)

És habitual que als formularis, s'afectin diversos elements CSS per exemple el **'label'**

```
.formInputText { padding: 15px 0 0; position: relative; width: 100px; }  
.formInputText > input { background: transparent; border: 0; border-bottom: 1px solid #d2d2d2; color: #212121;  
font-family: inherit; font-size: 16px; outline: 0; padding: 7px 0; transition: border-color 0.2s; width: 100%; }  
.formInputText > input::placeholder { color: transparent; }  
.formInputText > input:placeholder-shown ~ label { cursor: text; font-size: 16px; top: 20px; }  
.formInputText > label { pointer-events: none; }  
.formInputText > label,  
.formInputText > input:focus ~ label { color: #9b9b9b; display: block; font-size: 12px; position: absolute; top: 0;  
transition: 0.2s; }  
.formInputText > input:focus ~ label { color: rgb(0, 125, 255); }  
.formInputText > input:focus { border-bottom: 2px solid rgb(0, 125, 255); padding-bottom: 6px; }  
.formInputText > input:valid ~ span { color: transparent; }  
.formInputText > input:invalid ~ span { color: red; }
```

```
<div class="formInputText">  
  <input placeholder="yes" pattern="^[0-9]*" autocomplete="off" type="text"></input>  
  <label>Name</label>  
  <span>Numbers not allowed</span>  
</div>
```



Name

Numbers not allowed

Colors CSS

```
.c0 { color: rebeccapurple; }  
.c1 { color: #abc; }  
.c2 { color: #abcdef; }  
.c3 { color: rgb(100, 150, 200); }  
.c4 { color: rgba(100, 150, 200, 0.5); }
```

```
<p class="c0">Color de text</p>  
<p class="c1">Color hexadecimal de 3 xifres</p>  
<p class="c2">Color hexadecimal de 6 xifres</p>  
<p class="c3">Color rgb(vermell, verd, blau) entre 0 i 255</p>  
<p class="c4">Color rgba(vermell, verd, blau, alpha) amb  
transparència entre 0 i 1</p>
```

Color de text

Color hexadecimal de 3 xifres

Color hexadecimal de 6 xifres

Color rgb(vermell, verd, blau) entre 0 i 255

Color rgba(vermell, verd, blau, alpha) amb transparència entre 0 i 1

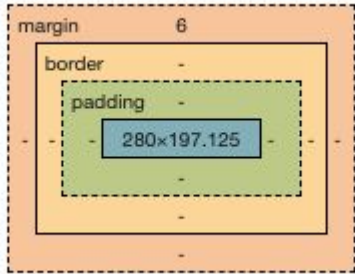
Maneres de definir colors:

- Per noms de color d'[aquesta llista](#)
- Amb # i 3 valors hexademimals: #abc
(#def equival a #ddeeff i #123 a #112233)
- Amb # i 6 valors hexadecimal: #abcdef
- Amb rgb(100, 150, 200) i valors vermell, verd i blau entre 0 i 255
- Amb rgba(100, 150, 200, 0.5) on l'últim valor és la transparència de 0 a 1

Mides CSS, envoltori dels elements

margin:

distància entre l'element i el què hi ha al seu voltant, (els margins es sobreposen)



border:

límits de l'element, es poden decorar amb línies i colors

padding:

distància entre l'element i els seus continguts

Mides CSS, unitats

```
#d0 {  
  background-color: aqua;  
  font-size: 0.75em;  
  height: 5%;  
  width: 100px;  
}  
#d1 {  
  background-color: lightsalmon;  
  font-size: 1.5em;  
  height: 5vh;  
  width: 50vw;  
}
```

```
<div id="d0">Capa 0</div>  
<div id="d1">Capa 1</div>
```

Normalment expressem les mides en:

px, pixels de la pantalla

%, percentatge relatiu a l'element pare

vw, relatiu a 1% de l'ample de la finestra

vh, relatiu a 1% de l'alt de la finestra

em, 1em equival a la mida de text configurada per l'usuari



Mides CSS, **box-sizing**

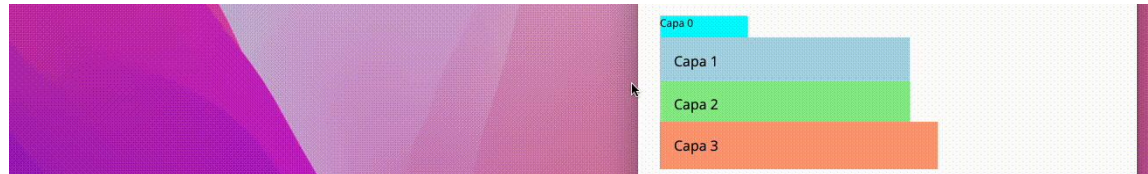
```
#d0 {  
  background-color: aqua;  
  font-size: 0.75em;  
  height: 25px;  
  width: 100px;  
}  
#d1 {  
  background-color: lightblue;  
  box-sizing: border-box;  
  height: 5vh;  
  padding: 16px;  
  width: 50vw;  
}  
#d2 {  
  background-color: lightgreen;  
  height: 15px;  
  padding: 16px;  
  width: calc(50vw - 32px);  
}  
#d3 {  
  background-color: lightsalmon;  
  padding: 16px;  
  width: 50vw;  
}
```

L'ample dels elements, per defecte no inclou el 'padding' ni el 'border'

Per fer que els inclogui hem de definir-ho al CSS amb **box-sizing: border-box;**

Veure la 'Capa 3' d'aquest exemple

```
<div id="d0">Capa 0</div>  
<div id="d1">Capa 1</div>  
<div id="d2">Capa 2</div>  
<div id="d3">Capa 3</div>
```



Mides CSS, funció calc

```
#d0 {  
  background-color: aqua;  
  font-size: 0.75em;  
  height: 25px;  
  width: 100px;  
}  
#d1 {  
  background-color: lightblue;  
  box-sizing: border-box;  
  height: 5vh;  
  padding: 16px;  
  width: 50vw;  
}  
#d2 {  
  background-color: lightgreen;  
  height: 15px;  
  padding: 16px;  
  width: calc(50vw - 32px);  
}  
#d3 {  
  background-color: lightsalmon;  
  padding: 16px;  
  width: 50vw;  
}
```

Amb la funció 'calc' podem calcular mides amb petites operacions matemàtiques sobre les mides relatives.

Va bé per restar el 'padding' i/o el 'border' a la mida si no fem servir box-sizing: "border-box"

```
<div id="d0">Capa 0</div>  
<div id="d1">Capa 1</div>  
<div id="d2">Capa 2</div>  
<div id="d3">Capa 3</div>
```



CSS segons la mida disponible amb **@media**

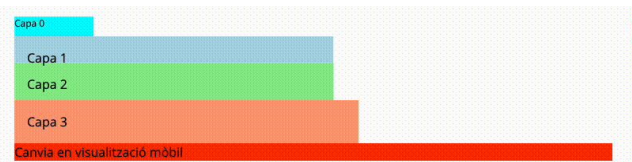
Perquè funcioni l'adaptació a diferents formats cal tenir limitat el viewport entre els 'meta' del head de l'arxiu .html, com en aquest exemple:

```
<meta name="viewport" content="initial-scale=1, maximum-scale=1" />
```

En aquest exemple, l'element amb atribut **data-css="css2"** té un color de fons vermell en pantalles més grans de 768px i verd en més petites

```
*[data-css="css2"] {  
  background-color: #ff0000;  
  font-family: "Open Sans";  
  margin: 0;  
  padding: 0;  
}  
@media only screen and (max-width: 768px) {  
  *[data-css="css2"] {  
    background-color: #00ff2a;  
  }  
}
```

```
<div id="d0">Capa 0</div>  
<div id="d1">Capa 1</div>  
<div id="d2">Capa 2</div>  
<div id="d3">Capa 3</div>  
<div data-css="css2">Canvia en visualització mòbil</div>
```



Transicions entre estats

```
<p class="quadre0">Quadre 0</p>  
<p class="quadre1">Quadre 1</p>
```

```
.quadre0 {  
  background-color: lightskyblue;  
  height: 50px;  
  margin-top: 50px;  
  transition: width 0.5s ease, background-color 1s linear 0.5s;  
  width: 100px;  
}  
.quadre0:hover {  
  cursor: pointer;  
}  
.quadre0:hover ~ .quadre1 {  
  background-color: red;  
  width: 200px;  
}  
.quadre1 {  
  background-color: burlywood;  
  transition: background-color 0.5s ease, width 1s ease-in-out;  
  width: 100px;  
}
```

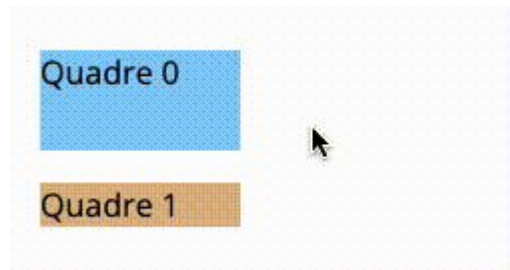
Les transicions apliquen una 'animació' entre dos estats diferents que pot tenir un mateix element CSS

Les transicions funcionen només per mides numèriques i colors



Transicions entre estats

```
.quadre1 {  
  background-color: burlywood;  
  transition: background-color 0.5s ease, width 1s ease-in-out;  
  width: 100px;  
}
```



Les transicions tenen diferents paràmetres:

```
transition: background-color 0.5s ease, width 1s ease-in-out;
```

- El nom de la propietat que canviarà
- El temps que trigarà a fer l'animació un cop canvia
- La [funció d'animació](#) que defineix com progressa l'animació amb el temps

Animacions vs Transicions

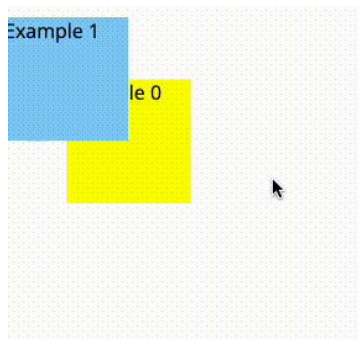
Les transicions defineixen com es mostra l'animació entre diferents estats dels elements CSS

Les animacions en canvi defineixen diferents estats de les propietats CSS en el temps

Animacions @keyframes

Per definir animacions es defineix el valor de les propietats en diferents moments amb **@keyframes**

- De from a to
- Per percentatge



```
.example0 {
  animation-name: framesExample0;
  animation-duration: 2s;
  animation-fill-mode: forwards;
  background-color: red;
  left: 0;
  height: 100px;
  position: absolute;
  top: 0;
  width: 100px;
}

@keyframes framesExample0 {
  from { background-color: red; top: 0; left: 0; }
  to   { background-color: yellow; top: 50px; left: 50px; }
}

.example1 {
  animation-name: framesExample1;
  animation-duration: 4s;
  background-color: lightskyblue;
  left: 0;
  height: 100px;
  position: absolute;
  top: 0;
  width: 100px;
}

@keyframes framesExample1 {
  0%   { background-color: lightskyblue; top: 100; left: 0; }
  33%  { background-color: lightsalmon; top: 150px; left: 50px; }
  66%  { background-color: lightgreen; top: 100px; left: 100px; }
  100% { background-color: lightskyblue; top: 100; left: 0; }
}
```

Variables

```
:root {  
  --color-principal: orange;  
}  
  
H1 {  
  color: var(--color-principal);  
}  
  
ion-icon {  
  color: green;  
}
```

Les variables permeten assignar un valor a un nom de variable, i fer-lo servir després al CSS

Combinat amb JavaScript permet canviar la visualització de mode clar a mode fosc fàcilment