CONTINGUTS

CSS Avançat: Introducció

A mesura que els projectes web es tornen més complexos, l'ús del CSS també ha d'evolucionar per oferir solucions més adaptatives, eficients i organitzades. El CSS avançat no només se centra en dissenyar una pàgina web atractiva, sinó també en assegurar-se que aquesta pàgina funcioni bé en diversos dispositius, sigui fàcil de mantenir i estigui optimitzada per a un rendiment òptim.

Importància del disseny adaptable i optimitzat

Un dels pilars del CSS avançat és el **disseny responsive**, que permet que les pàgines web s'adaptin a diferents mides de pantalla i dispositius. Aquesta tècnica és cada cop més important a causa de l'augment de l'ús de dispositius mòbils. Per aconseguir un disseny adaptatiu, el CSS avançat incorpora tècniques com:

1. **Media Queries avançade**
2. **Unitats relatives i flexibles**
3. **Layouts responsius amb Flexbox i Grid**

Millors pràctiques d'organització del CSS

A mesura que els projectes es fan més grans, és essencial mantenir una bona organització del codi CSS per evitar que es torni difícil de gestionar i mantenir. A continuació, algunes millors pràctiques per a l'organització del CSS en projectes avançats:

1. **Modularització del CSS**:
    a. **Dividir el codi CSS en fitxers més petits i temàtics**
    b. **Arquitectura CSS escalable (SMACSS, BEM)**
2. **Pre-processadors CSS**:
    a. **Sass** o **LESS**
3. **Optimització del CSS**:
    a. **Minificació del CSS**
    b. **Carregament asíncron del CSS**
4. **Utilització de frameworks CSS**:
    a. **Bootstrap**
    b. **Tailwind CSS**

CSS AVANÇAT

Selectors avançats en CSS

Els **selectors avançats** en CSS permeten aplicar estils amb molta més precisió i flexibilitat, permetent seleccionar elements específics en funció de la seva posició, relació amb altres elements o atributs.

1. Selectors d'atribut

Els selectors d'atribut permeten aplicar estils als elements HTML que contenen un determinat atribut o valor d'atribut. Són especialment útils per estilitzar elements de formularis o en casos en què utilitzis atributs personalitzats.

Exemples comuns:

Seleccionar elements amb un atribut específic:

```
input[type="text"] {
  border: 1px solid blue;
}
```

En aquest exemple, tots els elements <input> que tenen l'atribut type="text" tindran una vora blava.

Seleccionar elements amb un atribut que conté un valor específic:

```
a[href*="example"] {
  color: green;
}
```

Aquest exemple aplica el color verd a tots els enllaços (<a>) que contenen la paraula "example" en l'atribut href.

Seleccionar elements amb un valor d'atribut que comença amb un valor específic:

```
img[src^="https"] {
  border: 2px solid black;
}
```

Això aplica una vora negra a totes les imatges (<img>) que tenen una URL src que comença per "https".

Seleccionar tots els elements amb un atribut:

```
[personalitzat] {
  border: 1px solid red;
}
```

En aquest exemple, tots els elements que tenen l'atribut personalitzat tindran una vora vermella.

Selectors avançats en CSS

2. Selectors combinats

Els selectors combinats permeten seleccionar elements que tenen una relació específica amb altres elements dins del document. Els combinadors més comuns són:

a) Combinador de descendents ( )

Selecciona tots els elements que són descendents (fills, nets, etc.) d'un element pare.

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

Això aplica el color vermell a tots els paràgrafs (<p>) que són descendents de qualsevol element .

b) Combinador de fills directes (>)

Selecciona només els fills directes d'un element pare.

```
div > p {
  color: blue;
}
```

En aquest cas, només els paràgrafs que són fills immediats d'un tindran el text de color blau. Els nets o altres descendents no es veuran afectats.

c) Combinador de germans adjacents (+)

Selecciona l'element que està just després d'un altre element germà..

```
h1 + p {
  font-weight: bold;
}
```

Aquest exemple fa que el primer paràgraf (<p>) que segueix directament un títol (<h1>) es mostri en negreta.

d) Combinador de germans generals (~)

Selecciona tots els germans que segueixen un element determinat dins del mateix pare.

```
h2 ~ p {
  color: gray;
}
```

Aquest exemple fa que tots els paràgrafs (<p>) que segueixen un <h2> dins del mateix contenidor tinguin el text de color gris.

Selectors avançats en CSS

3. Selectors pseudo-classe

Les pseudo-classes són selectors que s'utilitzen per aplicar estils a un element en un estat o condició particular. Això permet canviar l'estil d'un element quan l'usuari hi interactua o quan l'element compleix certs criteris.

:hover: Aplica estils quan l'usuari passa el ratolí per sobre de l'element.

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

Aquest exemple canvia el color d'un enllaç a vermell quan l'usuari passa el cursor per sobre.

:first-of-type: Selecciona el primer element del seu tipus dins del seu pare.

```
p:first-of-type {
  font-weight: bold;
}
```

En aquest exemple, el primer paràgraf dins de cada element pare tindrà el text en negreta.

:nth-child(): Selecciona elements en funció de la seva posició entre els seus germans.

```
tr:nth-child(odd) {
  background-color: #f0f0f0;
}
```

Això aplica un color de fons gris clar a les files imparells d'una taula.

:not(): Selecciona tots els elements excepte aquells que coincideixin amb el selector especificat.

```
div:not(.important) {
  background-color: yellow;
}
```

- Això aplica un color de fons groc a tots els que no tenen la classe important.

Selectors avançats en CSS

4. Selectors pseudo-element

Els pseudo-elements permeten aplicar estils a una part específica d'un element, com el seu contingut abans o després, o una línia de text. Els pseudo-elements més comuns són **::before i ::after.**

::before: Afegeix contingut o estils abans del contingut real d'un element.

```
h1::before {
  content: "★ ";
  color: gold;
}
```

Aquest exemple afegeix una estrella daurada abans de cada <h1>.

::after: Afegeix contingut o estils després del contingut real d'un element.

```
p::after {
  content: " (Final)";
  color: gray;
}
```

Això afegeix la paraula "(Final)" en gris després de cada paràgraf.

CSS AVANÇAT

Selectors avançats en CSS

En aquest exemple:

- Els camps de text amb type="text" tenen una vora blava.
- Els paràgrafs fills directes de div tenen el text de color verd.
- Els enllaços es tornen vermells quan l'usuari passa el cursor per sobre.
- Els títols <h1> tenen una icona al davant gràcies a ::before.

```
<!DOCTYPE html>
<html lang="ca">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Selectors Avançats</title>
  <style>
    /* Selectors d'atribut */
    input[type="text"] {
      border: 2px solid blue;
    }

    /* Combinador de fills directes */
    div > p {
      color: green;
    }

    /* Pseudo-classe hover */
    a:hover {
      color: red;
    }

    /* Pseudo-element before */
    h1::before {
      content: " ◆ ";
    }
  </style>
</head>
<body>
  <h1>Selectors avançats en CSS</h1>
  <div>
    <p>Aquest paràgraf és fill directe d'un `div` i és de color verd.</p>
  </div>
  <input type="text" placeholder="Escriu alguna cosa">
  <a href="#">Enllaç amb hover vermell</a>
</body>
</html>
```

07

CSS AVANÇAT

Fonts i tipografia avançades

El control de la tipografia en una pàgina web és crucial per garantir una bona llegibilitat i una estètica consistent. En CSS avançat, tens accés a un conjunt de propietats que et permeten controlar diversos aspectes de les fonts, com la seva càrrega dinàmica, espaiat, pes, i línies de text. També explorarem com utilitzar fonts personalitzades amb la regla **@font-face**.

1. Tipografies web amb @font-face

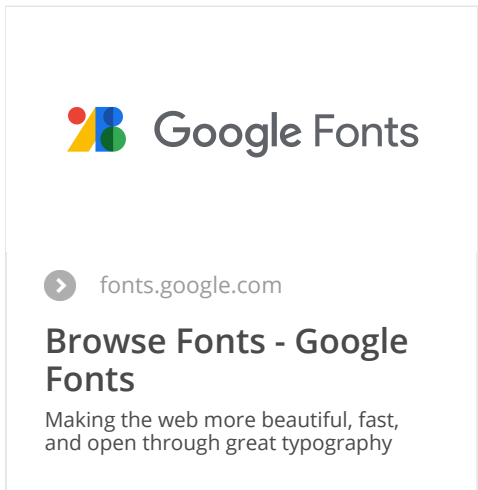
La regla @font-face permet utilitzar fonts personalitzades que no estan disponibles al sistema de l'usuari. Això significa que pots carregar fonts específiques des del teu servidor o des d'una font externa com Google Fonts, garantint una coherència tipogràfica entre tots els usuaris, independentment del dispositiu o navegador que utilitzin.

```
@font-face {
  font-family: "Open Sans";
  src: url("fonts/OpenSans-Regular.woff2") format("woff2"),
       url("fonts/OpenSans-Regular.woff") format("woff");
}

body {
  font-family: "Open Sans", sans-serif;
}
```

Fonts i tipografia avançades

Bancs de tipografies



Fonts i tipografia avançades

2. Control avançat de tipografia

A més de definir fonts personalitzades, CSS ofereix propietats avançades per controlar altres aspectes tipogràfics, com l'espai entre lletres, el gruix de la font o l'alçada de les línies.

a) Propietat letter-spacing

La propietat letter-spacing permet ajustar l'espai entre les lletres d'un text, fent que el text es vegi més comprimit o espaiat.

```
h1 {
  letter-spacing: 2px; /* Augmenta l'espai entre lletres */
}
```

b) Propietat line-height

La propietat line-height controla l'alçada de les línies de text. Una alçada de línia més gran augmenta l'espai entre les línies, cosa que pot millorar la llegibilitat.

```
p {
  line-height: 1.6; /* Augmenta l'alçada de les línies */
}
```

Fonts i tipografia avançades

c) Propietat font-weight
La propietat font-weight controla el gruix del text. A més dels valors **normal** i **bold**, es poden utilitzar valors numèrics entre **100** (molt prim) i **900** (molt gruixut), oferint més flexibilitat.

```
h1 {
  font-weight: 700; /* Gruix equivalent a negreta */
}
p {
  font-weight: 300; /* Gruix més prim */
}
```

d) Propietat font-style
La propietat font-style permet definir si el text es mostrarà en estil normal, cursiva o obliqua. És útil per donar èmfasi al text.

```
em {
  font-style: italic; /* Mostra el text en cursiva */
}
```

e) Propietat text-transform
Aquesta propietat permet canviar la capitalització del text, convertint-lo tot en majúscules, minúscules o capitalitzant la primera lletra de cada paraula.
uppercase: Converteix tot el text en majúscules.
lowercase: Converteix tot el text en minúscules.
capitalize: Capitalitza la primera lletra de cada paraula.

```
em {
  text-transform: uppercase;
}
```

Fonts i tipografia avançades

3. Ús del Web Open Font Format (WOFF)

WOFF i **WOFF2** són formats de font comprimits dissenyats específicament per a ús web. Aquests formats proporcionen una compressió més eficient que els formats tradicionals, millorant el temps de càrrega de les pàgines sense perdre qualitat tipogràfica. És recomanable utilitzar WOFF2 quan sigui possible, ja que és el format més modern i optimitzat.

```
@font-face {
  font-family: "Roboto";
  src: url("fonts/Roboto-Regular.woff2") format("woff2"),
       url("fonts/Roboto-Regular.woff") format("woff");
}
```

L'exemple carrega la font "Roboto" utilitzant els formats WOFF i WOFF2, garantint compatibilitat amb navegadors antics i moderns.

Fonts i tipografia avançades

Exemple complet de propietats avançades de fonts i tipografia:

En aquest exemple:

- Utilitzem la font personalitzada **Lato** carregada amb WOFF2 i WOFF.
- Les propietats letter-spacing, line-height, i font-weight s'utilitzen per controlar l'espaiat entre lletres, l'alçada de les línies i el gruix de la font.
- text-transform converteix el títol en majúscules.

```
<!DOCTYPE html>
<html lang="ca">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Propietats avançades de fonts</title>
  <style>
    @font-face {
      font-family: "Lato";
      src: url("fonts/Lato-Regular.woff2") format("woff2"),
           url("fonts/Lato-Regular.woff") format("woff");
    }

    body {
      font-family: "Lato", sans-serif;
      line-height: 1.6;
    }

    h1 {
      font-weight: 700;
      letter-spacing: 2px;
      text-transform: uppercase;
    }

    p {
      font-weight: 300;
      font-style: italic;
    }
  </style>
</head>
<body>
  <h1>Propietats avançades de fonts</h1>
  <p>Aquest és un paràgraf amb estil cursiva i un pes lleuger de font.</p>
</body>
</html>
```

Layouts amb Flexbox i Grid

Flexbox

Grid

Casos comuns per Flexbox:

1. Quan vols alinear elements en fila o columna, com botons en una barra de navegació o imatges en una galeria.
2. Quan vols distribuir l'espai entre elements en una fila (per exemple, centrant un element o espaiant-los uniformement).
3. Quan els elements han de créixer o reduir-se per omplir l'espai disponible, com en una barra lateral o una fila de cartes.

Casos comuns per CSS Grid:

1. Quan vols crear una pàgina amb una estructura complexa (com una pàgina amb capçalera, barra lateral, contingut principal i peu de pàgina).
2. Quan tens un disseny en format de quadrícula amb files i columnes, com un tauler de fotos o un disseny de cartes de productes.
3. Quan vols controlar exactament on va cada element en la quadrícula (pots col·locar elements en qualsevol cel·la de la quadrícula).

Layouts amb Flexbox i Grid

Quan es tracta de crear dissenys flexibles, adaptables i organitzats, Flexbox i CSS Grid són les dues eines més poderoses del CSS. Amb elles pots controlar la disposició dels elements en la pàgina, ja sigui en una fila, columna o en quadrícules complexes, adaptant els elements fàcilment a diferents mides de pantalla i requeriments.

1. Flexbox avançat: alineació, `flex-shrink`, `align-self`

Flexbox és una eina molt flexible per crear dissenys en fila o columna, on els elements s'alineen i ajusten automàticament segons l'espai disponible.

flex-shrink: Controla com es redueix la mida d'un element quan no hi ha prou espai disponible. Valors més grans permeten que l'element es redueixi més.

```
.element {
  flex-shrink: 1; /* Permet que l'element es redueixi quan no hi ha espai suficient */
}
```

align-self: Permet que un sol element en un contenidor flex tingui una alineació diferent a la resta d'elements. Els valors poden ser flex-start, flex-end, center, baseline o stretch.

```
.element {
  align-self: center; /* Centra aquest element individualment dins del contenidor */
}
```

Layouts amb Flexbox i Grid

2. Grid Layout: creació de quadrícules

CSS Grid Layout és una eina més potent que Flexbox quan es tracta de crear layouts en dues dimensions (filera i columna). Grid permet col·locar elements en una quadrícula amb molta precisió, controlant la mida de les files i columnes, i permetent que els elements s'ajustin a l'espai disponible.

Propietats clau en CSS Grid:

- **grid-template-columns i grid-template-rows**: Defineixen el nombre i la mida de les columnes i files dins del contenidor grid.
- **grid-column i grid-row**: Permeten que un element ocupi diverses columnes o files dins de la quadrícula.
- **gap**: Defineix l'espai entre les files i columnes de la quadrícula.

```css
.grid {
  display: grid;
  grid-template-columns: 1fr 2fr 1fr; /* Defineix 3 columnes amb mides flexibles */
  grid-template-rows: 100px auto; /* Defineix dues files: una fixa i una automàtica */
}
```

```css
.element-gran {
  grid-column: span 2; /* L'element ocuparà 2 columnes */
  grid-row: span 1; /* L'element ocuparà només 1 fila */
}
```

```css
.grid {
  gap: 20px; /* Defineix un espai de 20 píxels entre les files i columnes */
}
```

Layouts amb Flexbox i Grid

Exemple bàsic amb CSS Grid:

En aquest exemple:

1. La quadrícula té 3 columnes iguals.
2. L'element 2 ocupa dues columnes gràcies a grid-column: span 2.

```
.<!DOCTYPE html>
<html lang="ca">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Grid Bàsic</title>
  <style>
    .grid {
      display: grid;
      grid-template-columns: 1fr 1fr 1fr; /* 3 columnes iguals */
      grid-gap: 10px; /* Espai entre les columnes */
      height: 400px;
    }

    .element {
      background-color: lightblue;
      padding: 20px;
      text-align: center;
    }

    .element-gran {
      grid-column: span 2; /* Aquest element ocuparà dues columnes */
    }
  </style>
</head>
<body>
  <div class="grid">
    <div class="element">Element 1</div>
    <div class="element element-gran">Element 2 (Gran)</div>
    <div class="element">Element 3</div>
    <div class="element">Element 4</div>
  </div>
</body>
</html>
```

Layouts amb Flexbox i Grid

3. Ús combinat de Flexbox i Grid

Tot i que Flexbox i Grid es poden utilitzar per separat, sovint és útil combinar-los per obtenir el millor de tots dos. Per exemple, pots utilitzar **Grid** per definir la disposició general d'una pàgina (files i columnes) i utilitzar **Flexbox** dins d'algunes àrees per ajustar l'alineació dels elements.

En aquest exemple:

1. El **layout general** s'estructura amb **Grid**, dividint la pàgina en columnes.
2. Dins d'una de les cel·les del Grid, s'utilitza **Flexbox** per alinear els elements en una fila.

```html
<!DOCTYPE html>
<html lang="ca">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Flexbox i Grid combinats</title>
  <style>
    .grid {
      display: grid;
      grid-template-columns: 1fr 2fr;
      gap: 10px;
      height: 300px;
    }

    .element {
      background-color: lightblue;
      padding: 10px;
    }

    .flex {
      display: flex;
      justify-content: space-around; /* Flexbox dins del contenidor grid */
      align-items: center;
      background-color: lightgreen;
      height: 100%;
    }
  </style>
</head>
<body>
  <div class="grid">
    <div class="element">Element 1</div>
    <div class="flex">
      <div>Flex Element 1</div>
      <div>Flex Element 2</div>
    </div>
  </div>
</body>
</html>
```

Popover API

La **Popover API** és una funcionalitat recent que permet als desenvolupadors crear contingut emergent (popovers) de manera nativa, utilitzant només HTML i CSS, sense necessitat de JavaScript. Aquesta API facilita la creació de components com menús, suggeriments o notificacions que apareixen sobre el contingut existent.

Atributs HTML rellevants

popover: Atribut global que designa un element com a popover.

1. **"auto"** (per defecte): El popover es mostra i s'oculta automàticament.
2. **"manual"**: El control de la visibilitat del popover es gestiona manualment.

popovertarget: Atribut que especifica l'ID de l'element popover que es controlarà.

popovertargetaction: Defineix l'acció a realitzar sobre el popover.

1. **"show"**: Mostra el popover.
2. **"hide"**: Oculta el popover.
3. **"toggle"**: Alterna entre mostrar i ocultar el popover.

Popover API

Explicació de l'exemple:

1. El botó amb la classe trigger té l'atribut popovertarget que apunta a l'ID popoverContent, indicant que aquest botó controlarà la visibilitat del popover.
2. El div amb l'ID popoverContent té l'atribut popover, designant-lo com a contingut emergent.
3. Quan l'usuari fa clic al botó, el popover es mostra o s'oculta automàticament.
4. L'estil aplicat al pseudo-element ::backdrop crea un efecte de desenfocament al fons quan el popover està obert.


```html
<!DOCTYPE html>
<html lang="ca">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemple de Popover API</title>
  <style>
    /* Estils per al botó activador */
    .trigger {
      padding: 10px 20px;
      background-color: #3498db;
      color: white;
      border: none;
      cursor: pointer;
      border-radius: 4px;
    }

    /* Estils per al contingut del popover */
    #popoverContent {
      padding: 10px;
      background-color: #333;
      color: white;
      border-radius: 5px;
      width: 200px;
      text-align: center;
    }

    /* Estils per al fons quan el popover està obert */
    ::backdrop {
      backdrop-filter: blur(3px);
    }
  </style>
</head>
<body>
  <!-- Botó que activa el popover -->
  <button class="trigger" popovertarget="popoverContent">
    Mostra informació
  </button>

  <!-- Contingut del popover -->
  <div id="popoverContent" popover>
    <p>Aquest és un popover utilitzant la Popover API!</p>
  </div>
</body>
</html>
```

Variables CSS (Custom Properties)

Les **variables CSS**, també conegudes com **Custom Properties**, permeten definir valors reutilitzables dins d'un document CSS. Aquesta característica facilita molt la gestió i manteniment de grans fulls d'estil, ja que pots centralitzar valors com colors, mides i espaiats, i reutilitzar-los en diverses regles CSS. Si necessites fer un canvi en algun d'aquests valors, només hauràs de modificar la variable en un lloc, i el canvi es propagarà a totes les parts del document on s'utilitza.

1. Definir i utilitzar variables CSS

Les variables CSS es defineixen dins de qualsevol selector utilitzant el prefix --. És habitual definir-les en el selector :root per tal que estiguin disponibles globalment en tot el document.

```css
:root {
  --color-primari: #3498db;
  --marge-petit: 10px;
}

body {
  background-color: var(--color-primari);
  margin: var(--marge-petit);
}
```

Variables CSS (Custom Properties)

2. Beneficis de les variables en grans projectes

L'ús de variables és especialment útil en projectes grans o complexos, on necessites garantir la consistència en colors, mides, fonts i altres estils. Canviar aquests valors en múltiples llocs manualment pot ser tediós i propens a errors. Les variables centralitzen aquests valors en un sol lloc.

Avantatges:

1. **Mantenibilitat**: Pots modificar fàcilment valors globals com els colors o les mides canviant només una variable.
2. **Reutilització**: Les mateixes variables es poden utilitzar en qualsevol lloc del full d'estil, estalviant temps i espai.
3. **Consistència**: L'ús de variables assegura que tots els elements que depenen d'una variable tinguin el mateix estil.

Variables CSS (Custom Properties)

3. Variables CSS amb valors per defecte

Pots proporcionar un valor per defecte a les variables CSS en cas que no estiguin definides. Això és útil si vols assegurar-te que el disseny funcioni fins i tot si la variable falla o no es troba.

```
h2 {
  color: var(--color-primari, black); /* Utilitza negre si --color-primari no està definit */
}
```

Variables CSS (Custom Properties)

Exemple complet amb variables CSS:

En aquest exemple:

1. S'utilitzen variables CSS per gestionar els colors, la mida del text i els marges.
2. El botó canvia de color quan es passa el ratolí per sobre, utilitzant les mateixes variables per a la seva transició.

```
<!DOCTYPE html>
<html lang="ca">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Variables CSS</title>
  <style>
    :root {
      --color-primari: #3498db;
      --color-secundari: #2ecc71;
      --marge-general: 20px;
      --mida-text: 16px;
    }

    body {
      background-color: var(--color-primari);
      color: white;
      margin: var(--marge-general);
      font-size: var(--mida-text);
    }

    h1 {
      color: var(--color-secundari);
    }

    button {
      padding: 10px 20px;
      background-color: white;
      color: var(--color-primari);
      border: none;
      cursor: pointer;
    }

    button:hover {
      background-color: var(--color-secundari);
      color: white;
    }
  </style>
</head>
<body>
  <h1>Ús de Variables CSS</h1>
  <p>Aquest paràgraf utilitza variables per definir la mida del text, els colors i els marges.</p>
  <button>Botó interactiu</button>
</body>
</html>
```

Animacions i transicions en CSS

Les **animacions** i **transicions** en CSS són eines potents que permeten afegir moviment i interacció visual a les pàgines web, millorant l'experiència de l'usuari. Amb aquestes tècniques, pots controlar com els elements canvien les seves propietats (com color, posició o opacitat) de manera suau i fluida.

1. Ús de transicions (transition)

Les **transicions CSS** permeten que els canvis en les propietats d'un element, com el color, l'opacitat, la mida o la posició, siguin suaus en lloc de bruscos. Quan una propietat canvia, pots definir la velocitat del canvi utilitzant transition.

Sintaxi bàsica de transition:

```
element {
 transition: propietat durada funció-temporització retard;
}
```

1. **Propietat**: La propietat CSS que vols animar (ex. background-color, transform, opacity, etc.).
2. **Durada**: Quant de temps durarà la transició (ex. 0.5s, 2s).
3. **Funció de temporització**: Controla com serà la velocitat de la transició (ex. ease, linear, ease-in-out).
4. **Retard**: Opcional, defineix quant de temps trigarà a començar la transició (ex. 1s).

Exemple de transició simple:

```
button {
  background-color: lightblue;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: darkblue;
}
```

Animacions i transicions en CSS

2. Propietats de transició (transition-property)

Si vols animar diverses propietats simultàniament, pots especificar-les individualment amb transition-property o utilitzar all per aplicar la transició a totes les propietats que canviïn.

Exemple animant múltiples propietats:

```
div {
 width: 100px;
  height: 100px;
  background-color: lightgreen;
  transition: width 1s ease, background-color 0.5s ease-in-out;
}

div:hover {
  width: 200px;
  background-color: darkgreen;
}
```


En aquest exemple:

- Quan es passa el ratolí per sobre del div, l'amplada s'animarà durant 1 segon i el color de fons durant 0.5 segons, amb funcions de temporització diferents.

Animacions i transicions en CSS

3. Animacions clau amb @keyframes

Les animacions en CSS permeten crear moviments més complexos que les transicions. Utilitzen la regla @keyframes per definir punts intermedis del moviment o canvi de propietat, i s'apliquen a través de la propietat animation.

Sintaxi de @keyframes:

```
@keyframes nom-animacio {
  0% { propietat: valor-inicial; }
  100% { propietat: valor-final; }
}
```

Sintaxi de animation:

```
element {
  animation: nom-animacio durada funció-temporització retard cicles
      direcció;
}
```

Nom de l'animació: El nom de l'animació definida amb @keyframes.
Durada: El temps que durarà l'animació.
Funció de temporització: Controla la velocitat de l'animació (linear, ease, etc.).
Retard: Opcional, quant de temps trigarà a començar l'animació.
Cicles: Quantes vegades es repetirà l'animació (infinite per repetir-la indefinidament).
Direcció: Opcional, si l'animació es juga de manera normal o inversa (normal, reverse, alternate).

Animacions i transicions en CSS

```
@keyframes girar {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}

div {
  width: 100px;
  height: 100px;
  background-color: coral;
  animation: girar 2s linear infinite;
}
```

En aquest exemple:

- L'element gira contínuament gràcies a l'animació girar, que dura 2 segons i es repeteix indefinidament (infinite).

Animacions i transicions en CSS

4. Propietats d'animació (animation-duration, animation-delay)

Les propietats d'animació et permeten personalitzar com es comporta una animació. Les més importants són:

- **animation-duration**: Controla quant de temps dura l'animació.
- **animation-delay**: Defineix un retard abans que l'animació comenci.
- **animation-iteration-count**: Controla quantes vegades es repetirà l'animació (1, 3, infinite).
- **animation-direction**: Defineix si l'animació es juga cap endavant (normal), cap enrere (reverse) o alterna (alternate).

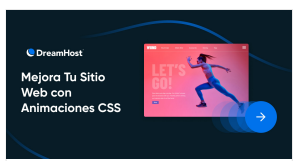
```css
/* Definició de l'element a animar */
.quadre {
  width: 100px;
  height: 100px;
  background-color: #3498db;
  position: relative;
  animation-name: moure;
  animation-duration: 2s;         /* Durada de l'animació */
  animation-delay: 1s;            /* Retard abans de començar */
  animation-iteration-count: 3;   /* Repetició de l'animació */
  animation-direction: alternate; /* Direcció alternada */
}

/* Definició de l'animació */
@keyframes moure {
  0% {
    left: 0;
  }
  100% {
    left: 200px;
  }
}
```

Animacions i transicions en CSS

En aquest exemple:

1. El botó canvia de color de manera suau quan es passa el ratolí per sobre gràcies a la transició.
2. El div es mou cap endavant i cap enrere de manera fluida gràcies a l'animació moure, que es repeteix indefinidament.



www.dreamhost.com

Animaciones CSS: Mejora Tu Sitio Web con Estas 17 Opciones

Una guía sobre cómo y por qué emplear animaciones CSS para mejorar tu experiencia de usuario. Además de 17 ejemplos CSS para tu sitio o aplicación.

Premis disseny web.

thefwa.com

Awards - The FWA

```html
<!DOCTYPE html>
<html lang="ca">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Animacions i Transicions en CSS</title>
  <style>
    /* Transició simple */
    button {
      padding: 10px 20px;
      background-color: lightblue;
      border: none;
      cursor: pointer;
      transition: background-color 0.3s ease;
    }

    button:hover {
      background-color: darkblue;
    }

    /* Animació clau */
    @keyframes moure {
      0% { transform: translateX(0); }
      50% { transform: translateX(100px); }
      100% { transform: translateX(0); }
    }

    div {
      width: 100px;
      height: 100px;
      background-color: coral;
      animation: moure 2s ease-in-out infinite;
    }
  </style>
</head>
<body>
  <button>Canvia color</button>
  <div></div>
</body>
</html>
```