**Stiluri pentru imagini**

* Opacity : 0-1; modifica transparenta, de la 0 invizibil la 1 normal
* Filter: functie(%), unde functie poaye fi:
* blur() – e in px nu %
* brightness()
* contrast()
* drop-shadow(x y blur color)
* grayscale()
* hue-rotate() – in deg, reverseaza culorile
* invert()
* opacity()
* saturate()
* sepia()

Putem aplica si mai multe filtre separate prin ,

* Griadientul permite sa inseram culori si sa facem o trecere de la o culoare la alta.

Este componenta din background-image

Avem 3 tipuri de gradient:

* Linear – se duce in sus/jos/stanga/dreapta/diagonal
* Radial – se imprastie pornind de la un centru
* Conic – rotit in jurul la un punct

1. **Linear**

**Background-image: linear-gradient(direction,color1,color2,..,colorN);**

Direction e optional, default fiind top to bottom

**Background-image: linear-gradient(to right, direction,color1,color2);**

Incepe de la stanga cu color1 spre dreapta cu color2

**Background-image: linear-gradient(to bottom right,color1,color2);**

Incepe de stanga sus si diagonal merge spre dreapta jos

.....

Directia din care incepe e mereu opusa celei ce noi o scriem, si putem pune orie directie

In loc de directie, putem folosi grade, unde 0deg e sus, 90 deg dreapta, 180 deg jos si 270 deg stanga .

1. **Radial**

**Background-image: radial-gradient(forma, color1, color2,..,colorN);**

Forma : circle, ellipse

Putem specifica cate % sa ia fiecare culoare

**Background-image: radial-gradient(forma, color1 10%, color2 5%,..,colorN);**

* **Transform: rotate/skew/scale/translate;**

Rotate(Xdeg) – roteste obiectul in spre coordonata

RotateX(deg) – roteste spre X, adica parca il rastoarna

RotateY(deg) – roteste spre Y, adica parca il intoarce

RotateZ(deg) – il roteste spre Z, adica il si rastoarna si intoarce

Rotate(x,y,z,unghi) – alegem toate 3

Rotate3d(x,y,z,unghi) x,y,z=[0,1]

Scale(nr) – mareste obiectul.Putem folosi valori cu – pentru a-l face ca in oglinda.

ScaleX(NR) – trage de obiect orizontal

ScaleY(NR) – trage de obiect vertical

ScaleZ(nr) – trage si vertical si orizontal

Skew(Xdeg, Ydeg) – denatureaza sau deformeaza obiectul

SkewX

skewY

* Transform: translate(x,y) – misca un obiect din pozitia sa initiala

Translate(x,y)

TranslateX()

Translate(Y)

* Transition-property: proprietate ;

Proprietatea arata care proprietate va fi afectata, de ex width, height,bakgroundcolor, color etc si aceasta trebuie definita mai sus

Transition-duration: 1s; - timpul cat va dura tranzitia

    background-color: #09A0FB;

    transition: background-color 1.5s;

poate fi scris si doar o transition. **Tranzitia va avea loc de indata ce intr-un :hover va fi definit un alt background-color**

div {

width: 100px;

height: 100px;

background: red;

transition-property: width;

transition-duration: 2s;

}

div:hover {

width: 300px;

}

* Transition-timing-function:
* Ease – incepe incet, dupa mai repede si la final iar incet
* Ease-in – incepe incet dupa repede
* Ease-out – incepe repede si incetineste
* Ease-in-out – inceputul si finalul sunt lente
* Linear – mereu e aceeasi viteza
* Step-start – trecerea se face brusc, fara animatie
* Step-end – la trecerea timpului, se face brusc animatia
* Steps – cu pasi
* Transition-delay: Xsec; amana animatia cu cateva secunda
* Pentru a face transizite la filter, trebuie sa punem dupa noul filtru setat pentru cand se pune mousul pe imagine:

Filter: inherit;

.Eiffel:hover{

    transition: filter 2s;

    filter: sepia(40%);

    filter: inherit;

}

**Animatii**

* Pentru a schimba anumite proprietati ale unui element in altele, folosim

@keyframes nume{

From { .. }

To { .. }

}

Unde in From specificam starea initiala, si apoi in To cea finala. De exemplu,

@keyframes animatie{

from{ background-color:red; }

to {background-color: green;

}

}

Va face ca un div, de ex, sa treaca de la imaginea red la green.

Apoi la acel div, de ex, folosim selectorii:

animation-name: numeAnimatie;

animation-duration: Timp in s;

De ex:

@keyframes animatie{

from{ background-color:red; }

to {background-color: green;

}

}

div{

width: 250px;

height: 250px;

animation-name: animatie;

animation-duration: 4s;

}

* Totusi, From si To seteaza de la 0% la 100%, insa putem sa specificam noi procentele:

@keyframes animatie{

0% {background-color: red; }

25% {background-color: green; }

50% {background-color: yellow; }

75% {background-color: black; }

100% {background-color: gray; }

}

div{

width: 250px;

height: 250px;

animation-name: animatie;

animation-duration: 4s;

}

* Pentru a o face sa se miste, folosim top, right, left, bottom

@keyframes animatie{

0% {left: 0px; top: 0px; }

25% { left: 200px; top:0px; }

50% { left:200px; top:200px }

75% { left:0px; top:200px}

100% { left:0px;top: 0px; }

}

div{

width: 250px;

height: 250px;

animation-name: animatie;

animation-duration: 8s;

background-color: red;

position: relative;

}

Px sunt ca coordonatele.

* animation-delay – intarzie animatia

Poate lua valori negative, si ar fi ca si cum animatia deja ar fi inceput de la acea perioada indicata.

* Animation-iteration-count: arata de cate ori animatia trebuie sa se repete.

Putem pune si **infinite**

* Animation-direction:
* Normal
* Reverse – se va misca invers
* Alternate – merge inainte, apoi invers apoi inainte si apoi invers si tot asa
* Alternate-reverse – merge inapoi apoi inainte si tot asa
* Animation-timing-function: identic ca la transition
* animation-fill-mode property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).

The animation-fill-mode property can have the following values:

* none - Default value. Animation will not apply any styles to the element before or after it is executing
* forwards - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
* backwards - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
* both -