1. Dynamic elements on the web

CSS animations allow you to define a set of CSS property changes that are applied sequentially over a period of time.

CSS transitions and animations are useful for adding dynamism to the user interface, such as button color changes when the user hovers over them, smooth cursor movements when selecting an interface element, or more complex animations to to elements such as charts and slides. This can improve the user experience and make the website more visually appealing.

1.1. CSS transitions

CSS transitions are a feature that allows changes to an element's properties to occur smoothly and progressively rather than instantaneously.

To use CSS transitions, you must first specify which CSS properties should have a transition, and also the duration of the transition.

This property has values:

1. The name of the CSS property to animate (ex background for the background property or font-size for the font size): transition-property.
2. The time for the transition to complete (in seconds or milliseconds): transition-duration.
3. The acceleration function that determines how the transition evolves over time: transition-timing-function.

```
button {

  transition: color 0.5s ease;

}

 button:hover {

  color: red;

}
```

In this example, the transition will be applied to the color property of the button, and will take 0.5 seconds to complete with a slow speed

Transitions can also be applied to multiple CSS properties at the same time, separating each property and its values with a comma. For example:

```
button {

  transition: color 0.5s ease, background 0.3s linear, font-size
0.2s ease-in;

}

 button:hover {

  color: red;

  background: green;

  font-size: 18px;

}
```

In this example, the transition will be applied to the color, background, and font-size property of the button at the same time, with different time, speed, and acceleration function.

Function	Effect
ease	It starts and ends slowly.
ease-in	Start slowly.
ease-out	It ends little by little.
lineal	There is no acceleration.

These are not all the features available, there are more.

To achieve a gradual change in the size of an element when the user hovers over it, the following code can be used:

```

/* Define the size property transition. */

.item {

  transition: width 0.5s, height 0.5s;

}

/* When the user hovers over it, increase the size of the element. */

.element:hover {

  width: 200px;

  height: 100px;

}
```

In this example, a transition has been defined for the width and height properties of the element. When the user hovers over the element, its width and height increase gradually

You can also play with the opacity of an element, applying a fadeIn-fadeOut effect to hide or show the element.

```
/* Define the transition of the opacity property. */

.item {

  transition: opacity 0.5s;

}

/* When the element is displayed, the opacity starts at 0 and
increases to 1. */

.element.show {

  opacity: 1;

}

/* When the element is hidden, the opacity starts at 1 and decreases
until it reaches 0. */

.element.hide {

  opacity: 0;

}
```

When the element is displayed, its opacity gradually increases from 0 to 1, creating a fadeIn effect. When the element is hidden, its opacity gradually decreases from 1 to 0, creating a fadeOut effect.

A transform function can be used to apply effects such as rotating elements.

```
/* Define the transformation property transition. */

.item {

  transition: transform 0.5s;

}

/* When the user hovers over it, the element rotates 45 degrees. */

.element:hover {

  transform: rotate(45deg);

}
```

When the user hovers the mouse over the element, it rotates 45 degrees smoothly instead of with a sudden change.

The transform CSS property has several possible values that allow different transformations to be applied to an element, such as rotation, scaling, translation, and warping.

Function	Effect
rotate(angle)	Rotates the element by a specified angle in degrees.
scale(factor)	Scales the element by a given factor.
translate(x, y)	Move the element by a specified distance in the x and y coordinates.
skew(x-angle, y-angle)	Deforms the element by a specified angle in degrees in the x and y coordinates.

The transform property can also be used with combined values

```
transform: rotate(45deg) scale(1.5) translate(50px, 100px) skew(10deg, 20deg);
```

This would rotate the element 45 degrees, scale it by a factor of 1.5, move it 50 pixels in the x direction and 100 pixels in the y direction, and finally warp it 10 degrees in the x direction and 20 degrees in the y direction.

The filter property, allows you to apply different special filters to the element such as erasing the image, increasing the contrast, adding shadows, changing the color image to black and white or applying a sepia color, among others.

```
img {

  transition: all 1s ease;

  display: block;

  margin: 20px;

  float: left;

}

.un:hover {

  filter: blur(5px);

}

.dos:hover {

  filter: grayscale(100%);

}

.tres:hover {

  filter: sepia(100%);
}
.quatre:hover {
  filter: contrast(4);
}


.cinc:hover {
  filter: hue-rotate(120deg); rotació de to de color
}
.sis:hover {
  filter: invert(1);
}
```

Function	Effect	Example
grayscale()	Convert the element to grayscale.	filter:grayscale(100%)
blur()	difumina	filter:blur(5px)
sepia()	Tints the element sepia.	filter:sepia(100%)
contrast()	Increases the contrast of the element.	filter:contrast(4)
invert()	Inverts the colors of the element.	filter:invert(1)

- menu.html
- imatges.html

1.2. CSS animations

CSS animations are created using the CSS animation property and various subproperties such as animation-duration, animation-timing-function, and others. These properties allow you to specify the different aspects of the animation, such as duration, transition type, and repeat (or not). This is achieved by writing a CSS rule that applies the animation property to the HTML element.

The @keyframes directive is then used to define the different states of the animation in the form of keyframes. Each keyframe specifies how the element should look at a certain point in the animation, and is defined using a CSS rule that specifies which CSS properties should change and how they should change.

When the web page loads, the browser executes the animation by causing the element to progressively change from one state to another according to the sequence of keyframes defined in the @keyframes directive.

- Animacio.html

The position property, with the relative value, is required to apply transformations (such as translateX) to an element. This is because transformations are applied relative to the element's current position, and if a relative position is not set, the element will not have a position set to apply the transformation to.

The rule for the animation class states that the animation is called "move-left" and lasts five seconds, with a linear interpolation.

The animation sequence is defined using the @keyframes directive. This sequence specifies that the square starts from its original position (no translation) and ends up moving to the left out of the viewport (-100%).

Finally, the document includes a <div> element that uses the square and move-left classes to apply the appearance and animation to the pink square.

There are many different ways to use CSS animations.

A simple example is for an element to rotate on an axis:

- Gir.html
- Mida.html

CSS animations can also be more complex and use different transform properties and style changes to create different effects.

- Midaimoure.html

Property	Description
Animation-name	Sets the name of the animation to which the element is applied. An animation defined with the @keyframes property must be referenced.

Animation-duration	Set the duration of the animation in seconds or milliseconds. For example, 2s or 500ms.
Animation-timing-function	Sets the time function used to calculate the values of the animation during its cycle. Some common options are: linear, ease, ease-in, ease-out and ease-in-out.
Animation-delay	Set the time to wait before the animation starts.
Animation-iteration-count	Set the number of times the animation is repeated. You can set a fixed number or the value "infinite" to make the animation repeat indefinitely.
Animation-direction	Set the direction of the animation. Possible values are: normal, reverse, alternate and alternate-reverse.
Animation-play-state	Set the playback state of the animation. Possible values are: running and paused.
Animation-fill-mode	Set how the animation behaves when it is not playing. Possible values are: none, forwards, backwards and both.
animation	It is the shorthand syntax for setting all animation properties on a single line. The values must be set in the following order: name duration timing-function delay iteration-count direction fill-mode.

For example, to apply all these properties to an element on a web page

- Animaciotots.html

This example has created an animation called "movelCanviarMida" that causes a square to move from right to left and increase its size at the same time for 2 seconds with an ease-in time function. The animation waits for 1 second before starting, repeats twice, and has an alternate direction (alternates between the "normal" and "reverse" direction on each iteration) and a backwards refill mode (the animation remains in its final state when not in playback). So when the web page is opened you will see a red square that moves from right to left and increases in size for 2 seconds, waits for 1 second before starting, repeats twice and has an alternate direction.

- Animaciotots2.html

In this example, two animations called "rotate" and "mourelCanviar" have been created that make a rectangle rotate 360 degrees for 3 seconds with a linear and ease-in-out time function respectively. The "movelCanviar" animation causes the circle to move from right to left, increase its size, and rotate 360 degrees for 3 seconds with an ease-in-out time function. Also, halfway through the animation, the circle changes color to yellow. The animation repeats indefinitely and has an alternate direction (alternates between normal and reverse direction at each iteration). You will also see a green rectangle that rotates 360 degrees for 3 seconds, waits 1 second before starting, repeats indefinitely, and has an alternate-reverse direction.

Example of images with animations.

- Portfolio.html