STUDENT NAME	**R.ROJA**	
STUDENT REGISTRATION NUMBER	**251U1R2064**	**CLASS: CSE(AIML)**
PROGRAM	**UG**	**YEAR and TERM: 1st year & 1st term**
SUBJECT NAME	**HTML**	
NAME OF THE ASSESSMENT	**Video concept**	
DATE OF SUBMISSION	**2.11.25**	

How to define animations using key frames:

In CSS, you define keyframes using the @keyframes rule:

```
@keyframes animation Name {
  0% {
    /* Starting state */
    transform: translate X(0);
    opacity: 0;
  }
  50% {
    /* Middle state */
    transform: translate X(50px);
    opacity: 0.5;
  }
  100% {
    /* Ending state */
    transform: translate X(100px);
opacity: 1;
  }
}
```

Explanation:

Animation Name → Name of your animation, used later to apply it.

Percentages (0%, 50%, 100%) → Define the timeline of the animation.

Inside each block, you define the CSS properties to animate.

2.Applying the Animation to an Element

Once the keyframes are defined, use the animation property on your element:

```
.box {
  width: 100px;
  height: 100px;
  background-colour: red;

  /* Apply the animation */
  animation: animation Name 3s ease-in-out infinite;
}
```

Animation Shorthand Breakdown:

- Animation Name → Refers to the keyframes you defined.
- 3s → Duration of the animation.
- ease-in-out → Timing function for speed progression.
- infinite → Repeat infinitely.

3.Tips for smooth transitions and peformance friendly animations

- **Animate transform and opacity** instead of layout properties like `width`, `height`, or `top`, since these trigger **repaints and reflows** which are costly.
- **Use `will-change` sparingly** to hint the browser about upcoming changes, helping it optimize rendering.
- **Prefer `request Animation Frame`** for JavaScript animations instead of `set Timeout` or `set Interval`, ensuring smoother frame updates.
- **Limit heavy effects** like box shadows, filters, or large images during animations—they can slow down the GPU.
- **Keep animations short and simple**, and avoid animating too many elements at once to prevent frame drops.