COESIONE ITALIA 21-27 PIEMONTE

Cofinanziato dall'Unione europea

REGIONE PIEMONTE

Web Developer

HTML, CSS e Strumenti di Digital Marketing (SEO, SEM, SEA)

Docente: Shadi Lahham

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Shadows, gradients & animations

Decorative effects

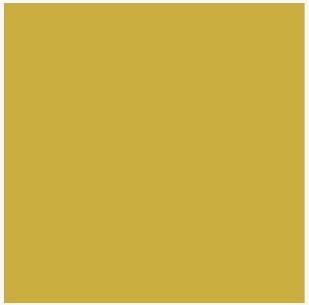
Shadi Lahham - Web development

Shadows

Box shadow

used to draw shadow effects around an element

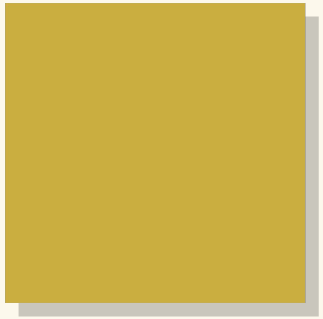
```
.box {
  width: 200px;
  height: 200px;
}

.box.ocra {
  background-color: #caae3f;
}
```



Box shadow

can have X and Y offsets, blur, spread radius, and color

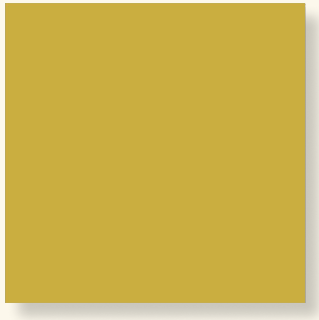
```
.shadow {
  box-shadow: 9px 9px rgba(0, 0, 0, 0.2);
}
```



Box shadow

can have X and Y offsets, blur, spread radius, and color

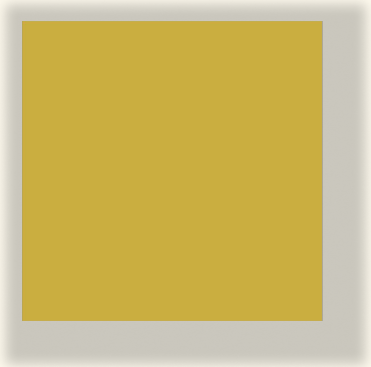
```
.shadow-blur {
  box-shadow: 9px 9px 9px rgba(0, 0, 0, 0.2);
}
```



Box shadow

can have X and Y offsets, blur, spread radius, and color

```
.shadow-spread {
  box-shadow: 9px 9px 9px 20px rgba(0, 0, 0,0.2);
}
```

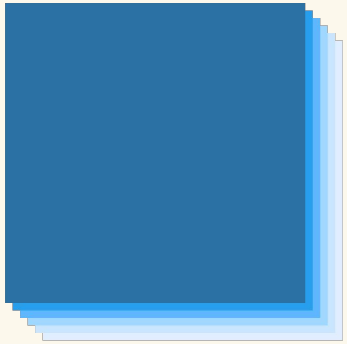


Box shadow

multiple shadows can be used at the same time on the same element

```
.box.blue {
  background-color: #2b71a3;
}

.shadow-tail {
  box-shadow: 5px 5px 0px 0px #289fed,
   10px 10px 0px 0px #5fb8ff,
   15px 15px 0px 0px #a1d8ff,
   20px 20px 0px 0px #cae6ff,
   25px 25px 0px 0px #e1eeff,
   5px 5px 15px 5px rgba(0, 0, 0, 0);
}
```

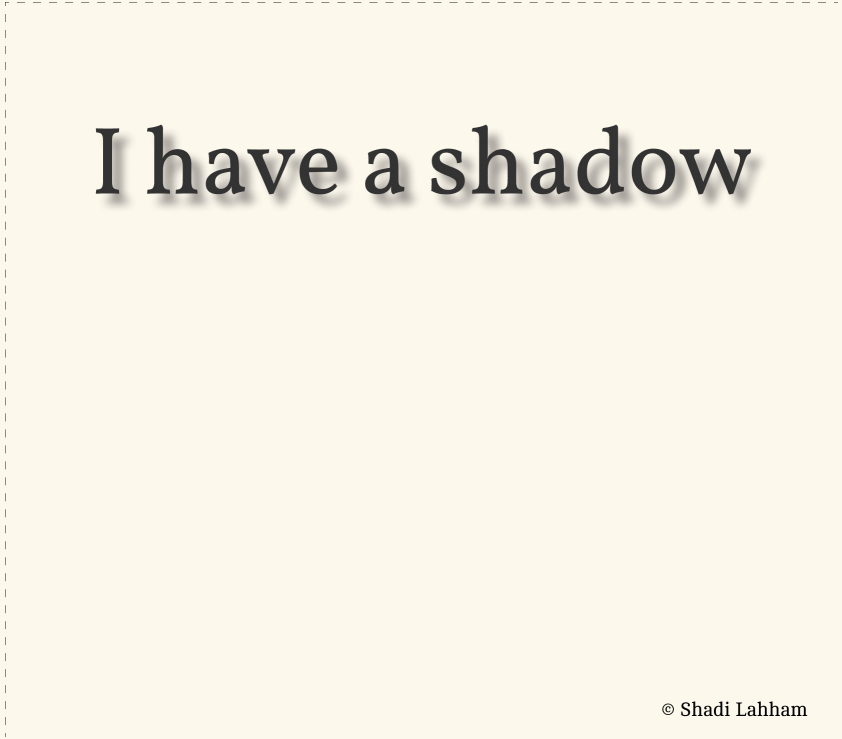
box-shadow | MDN



Text shadow

adds shadows to text
can have X and Y offsets, blur, and color

```
.shadow-text {
  font-size: 40px;
  text-shadow: 4px 4px 4px rgba(82, 77, 77, 0.6);
}
```



Text shadow

multiple shadows can be used at the same time on the same text

```
.shadow-multi-text {
  font-size: 40px;
  color: #2b71a3;
  text-shadow: 5px 5px 0px #289fed,
   10px 10px 0px #5fb8ff,
   15px 15px 0px #a1d8ff,
   20px 20px 0px #cae6ff,
   25px 25px 0px #e1eeff,
   5px 5px 15px rgba(0, 0, 0, 0);
}
```

text-shadow | MDN

Gradients

Gradients

transition between two or more colors, usually used as a background

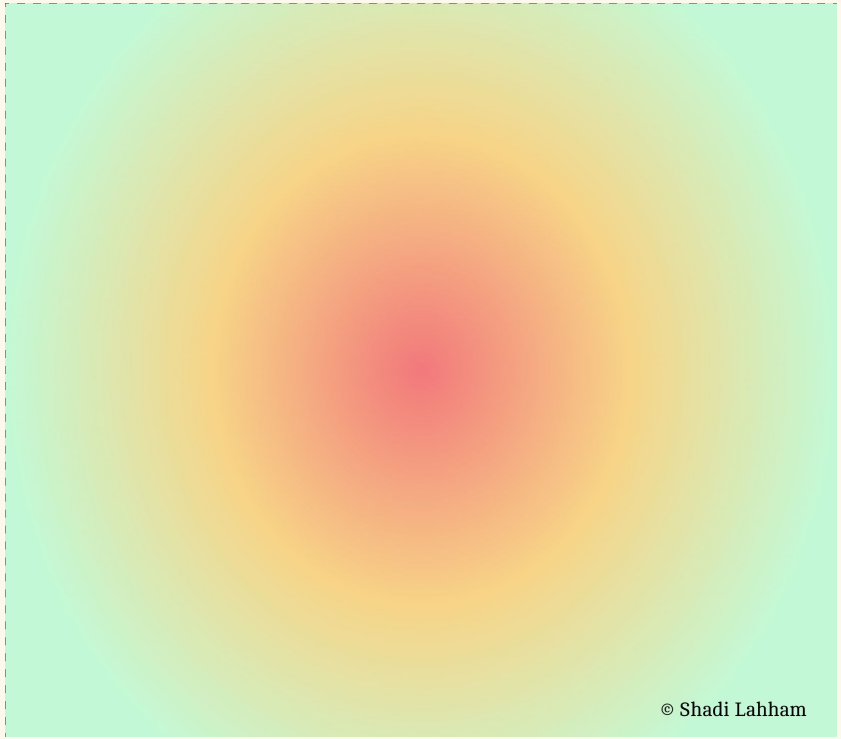
```
background: linear-gradient(
  90deg,
  hsla(302, 82%, 76%, 1) 0%,
  hsla(258, 40%, 68%, 1) 100%
);
```

linear-gradient() | MDN

Gradients

```
could be either linear or radial

background: radial-gradient(
  circle,
  hsla(358, 82%, 71%, 1) 0%,
  hsla(41, 88%, 75%, 1) 50%,
  hsla(141, 81%, 87%, 1) 100%
);
```

radial-gradient() | MDN



Gradients

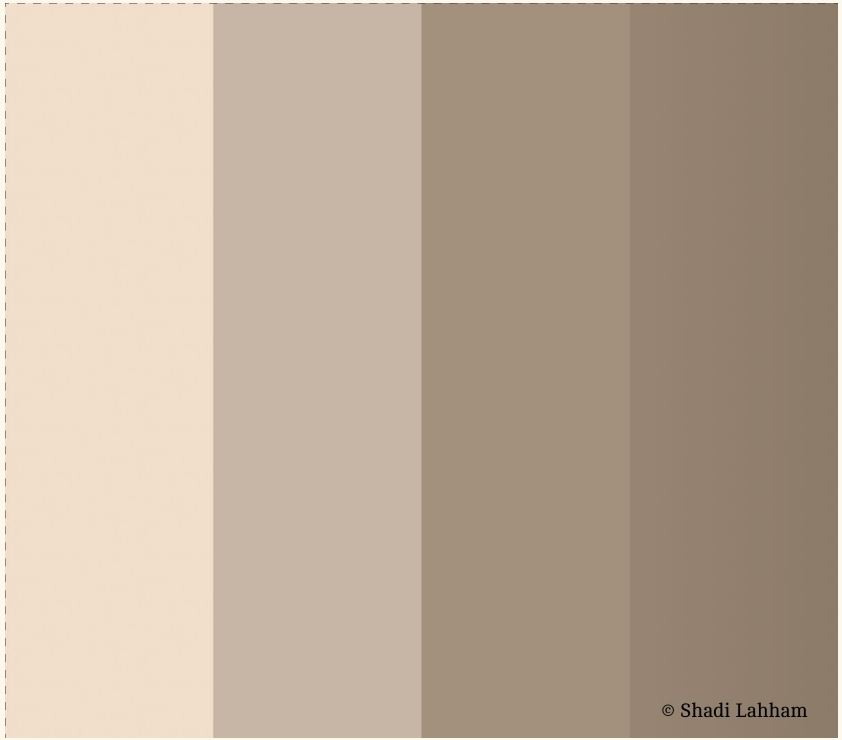
using multiple and transparent colors

```
background: linear-gradient(
  135deg,
  rgba(247, 89, 49, 1) 0%,
  rgba(233, 109, 94, 1) 62%,
  rgba(232, 111, 98, 0.7) 68%,
  rgba(224, 121, 121, 0) 100%
);
```

Gradient Maker
uiGradients
CSS Gradient Generator

Gradients

```
:root {
  --color-1: #f1dec9;
  --color-2: #c8b6a6;
  --color-3: #a4907c;
  --color-4: #8d7b68;
}

body {
  background: linear-gradient(
    90deg,
    var(--color-1) 0%,
    var(--color-1) 25%,
    var(--color-2) 25%,
    var(--color-2) 50%,
    var(--color-3) 50%,
    var(--color-3) 75%,
    var(--color-4) 75%,
    var(--color-4) 100%
  );
}
```



Transitions

Transition property

The CSS transition property is a powerful feature in CSS that allows you to control the intermediate steps in a CSS property change, creating smooth and gradual transitions between different states of an element

It is commonly used to enhance user experience by providing visual feedback when an element changes state, such as when it is hovered over, focused, or activated

Transition shorthand

the transition property is shorthand for four sub-properties

transition-property **(default all)**
the CSS property or properties to which the transition effect will be applied

transition-duration **(default 0s)**
the duration over which the transition takes place, specified in seconds or milliseconds

transition-timing-function **(default ease)**
how intermediate values of the transition are calculated
e.g. linear, ease, ease-in, ease-out, etc.

transition-delay **(default 0s)**
specifies the time to wait before starting the transition, also specified in seconds or milliseconds

example

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

Transition example

```css
div {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition: background-color 0.5s ease-in-out;
}

div:hover {
  background-color: green;
}
```

Multiple transitions

```css
/*  multiple transitions to animate size, shape (border-radius), and color on hover */
div {
  width: 100px;
  height: 100px;
  background-color: #dba34e;
  border-radius: 0;
  transition: width 1s ease-in-out, height 2s ease-in, background-color 0.5s ease-out,
    border-radius 1s ease-in-out;
}

div:hover {
  width: 200px;
  height: 200px;
  background-color: #ad60f5;
  border-radius: 8px;
}
```

Multiple transitions

Multiple transitions are beneficial for creating complex animations and enhancing user experience

However, excessive use can lead to distraction and performance issues

Use them wisely to avoid overwhelming users and maintain interface clarity

CSS transitions and hover animations, an interactive guide

Transformations

CSS transformations

Let developers manipulate the appearance of webpage elements dynamically to enable changes in position, size, and orientation without affecting the document flow

Through properties like `transform`, `translate`, `rotate`, and `scale`, developers can create visually engaging effects and animations effortlessly

They are crucial for crafting responsive designs, enhancing user interactions, and adding depth to web content, enabling the creation of captivating and interactive web experiences

Transformation properties

transform
main property used for applying transformations: translate, rotate, scale, skew, etc.

translate
moves an element from its current position (along the X and Y axes)

scale
increases or decreases the size of an element

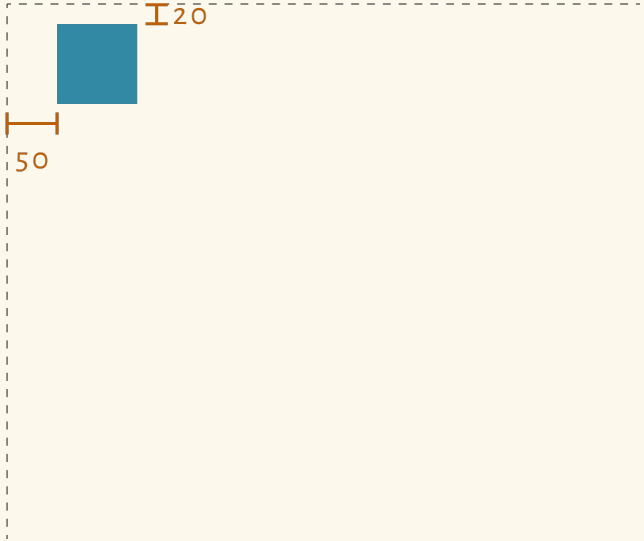
rotate
rotates an element by a specified angle

skew
skews an element along the X and Y axes

transform-origin
defines the point around which transformations are applied

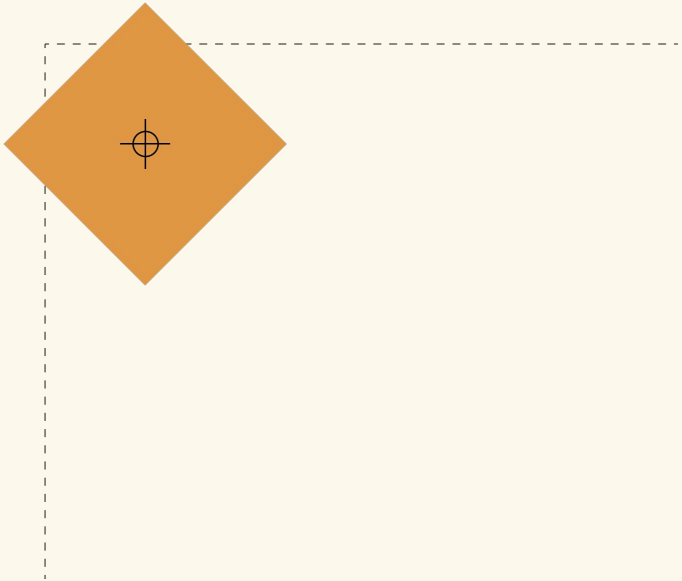
Translate

```css
.box {
  width: 80px;
  height: 80px;
  background-color: #3189a4;
  /* added for smooth animation */
  transition: transform 0.3s ease-in-out;
}

.box:hover {
  /* move box 50px to the right and 20px down */
  transform: translate(50px, 20px);
}
```



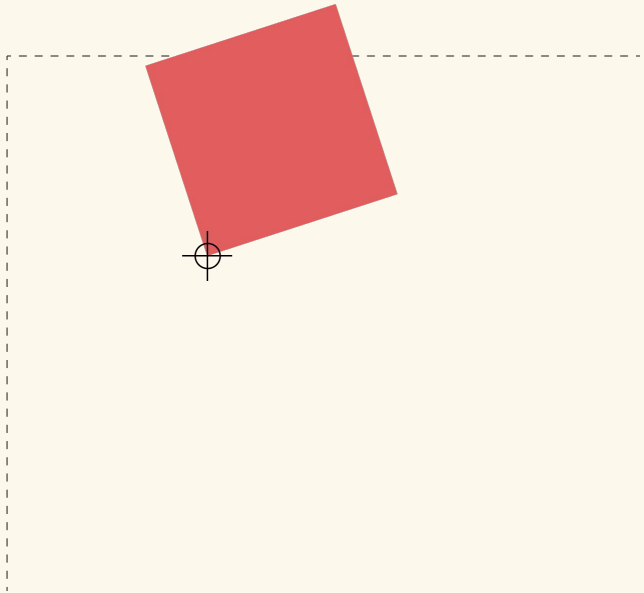
Rotate

```css
.photo {
  width: 200px;
  height: 200px;
  background-color: #e09743;
  transition: transform 0.5s ease-in;
}

.photo:hover {
  /* rotate image by 45 degrees on hover */
  transform: rotate(45deg);
}
```



Rotate & transform-origin

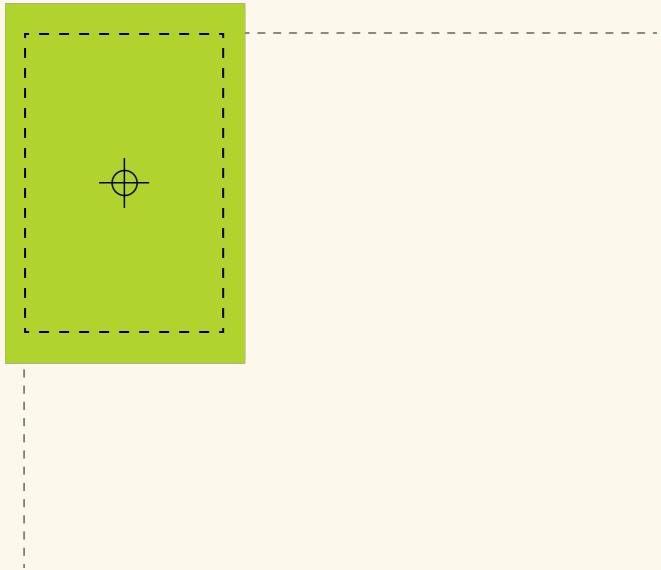
```css
.rotating-image {
  width: 200px;
  height: 200px;
  background-color: #e25d5d;
  transition: transform 0.3s ease;
  /* set bottom-right as point of reference */
  transform-origin: bottom right;
}

.rotating-image:hover {
  /* rotate image by 72 degrees on hover */
  transform: rotate(0.2turn);
}
```



There is a turn unit in CSS

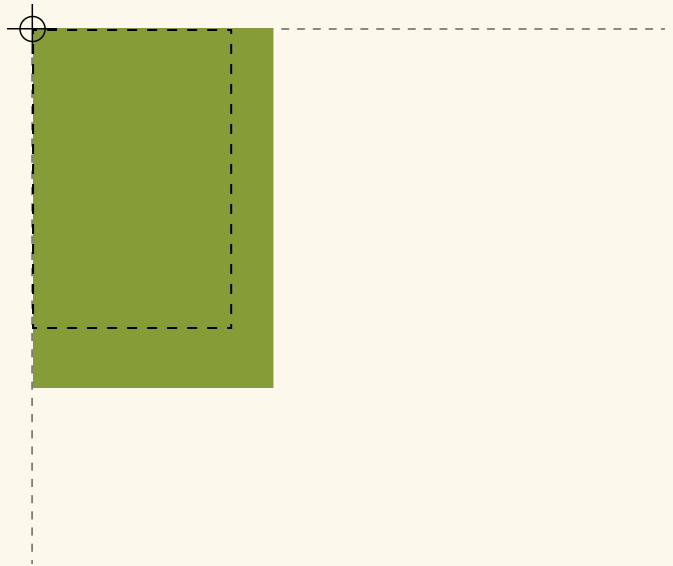
Scale

```
.zoom-image {
  width: 100px;
  height: 150px;
  background-color: #b0d42d;
  transition: transform 250ms linear;
}

.zoom-image:hover {
  /* scale image by 20% on hover */
  transform: scale(1.2);
}
```



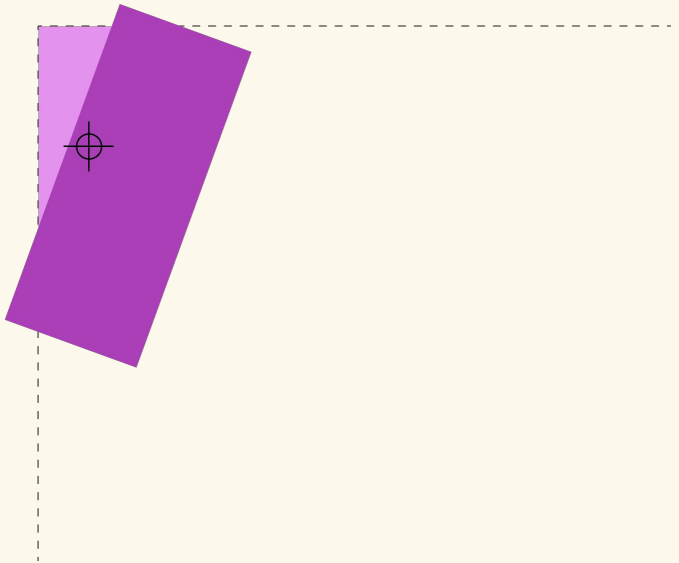
Scale & transform-origin

```css
.zoom-image.alt {
  width: 100px;
  height: 150px;
  background-color: #869c36;
  /* set top-left as point of reference */
  transform-origin: top left;
  transition: transform 250ms linear;
}

.zoom-image:hover {
  /* scale image by 20% on hover */
  transform: scale(1.2);
}
```



Multiple transformations

```
.transformed-image {
  width: 100px;
  height: 240px;
  background-color: #aa3fb8;
  transition: transform 0.4s ease-out;
}

.transformed-image:hover {
  /* translates, rotates, and scales the image */
  transform: translate(40px, 40px) rotate(20deg)
scale(1.4);
}
```



Animations

Keyframe animations

A powerful tool for creating dynamic and engaging web experiences by defining the intermediate steps, keyframes, between the initial and final states of an element's styling properties, such as position, size, color, and opacity

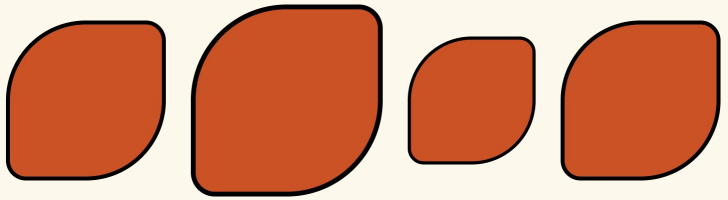
Designers can orchestrate smooth transitions and complex movements, adding visual interest and interactivity without JavaScript or external libraries

By specifying keyframes at various percentages of an animation's duration, developers can achieve precise control over the timing and behavior of elements, enhancing the overall user experience

Keyframe animations

```css
/* define animation */
@keyframes beat {
  0% { transform: scale(1); }
  25% { transform: scale(1.1); }
  50% { transform: scale(0.9); }
  100% { transform: scale(1); }
}

.shape {
  width: 80px;
  height: 80px;
  background-color: #ca5225;
  border: 2px solid black;
  border-radius: 50% 10px;
  transition: transform 0.5s;
}
```

```css
/* trigger animation on hover */
.shape:hover {
  animation: beat 2s infinite;
}
```



© Shadi Lahham

[animation | MDN]

Keyframe animation generators

Keyframe animation generators simplify the process of creating CSS animations by providing user-friendly interfaces to visually design animations, saving time and effort in writing complex CSS code manually

Animista
Keyframe Animation CSS Generator
CSS Animation Kit

Animations

CSS vs Javascript

User-perceived performance

Web animations can be created using JavaScript and CSS, each with its own pros and cons, and historically, web animations have evolved from simple effects to complex interactions, so choosing between JavaScript and CSS depends on factors like ease of implementation and performance

User-perceived performance is crucial as users interact with systems through various senses; therefore, animations should be optimized to provide responsive and smooth experiences, and while both methods are valid, code that prioritizes UPP will always perform better

CSS animation advantages

Simplicity and Ease of Use

Require less code than JavaScript

Performance

More performant due to GPU usage for smoother rendering

Browser Optimization

Modern browsers optimize them for efficiency

CSS animation disadvantages

Limited Control

Less flexible with limited control over animation flow and complex logic

Less Interactivity

Less dynamic handling of user input compared to JavaScript animations

JavaScript animation advantages

Control and Flexibility
Allows fine-grained control, complex sequences, conditional logic, and user input interactions

Complex Calculations
Suitable for real-time data-dependent or intricate logic animations

Event Handling
Integrates with events for interactive animations responding to clicks, hovers, and scrolls

JavaScript animation disadvantages

Performance

Less performant than CSS on low-power devices, potentially leading to jankiness

Complexity

More complex and verbose than CSS

Recommended

Use CSS for simple animations such as transitions and hover effects, as well as those that do not necessitate complex logic or interaction

CSS animations are efficient and straightforward to implement, making them particularly well-suited for performance-critical applications thanks to GPU acceleration and execution off the main thread

Less Recommended

Use JavaScript to handle complex animations that include intricate sequences, real-time data, or heavy interactivity, ensuring to exercise caution to prevent performance issues

JavaScript plays a crucial role in animations that react to intricate user interactions or synchronize with dynamic content

Historical Perspective

JS & CSS animations

In the early days of the web, animations were primarily done using JavaScript, often with libraries like jQuery for cross-browser compatibility which involved manually manipulating the DOM and CSS properties, which could be cumbersome and lead to performance issues

As web standards evolved, many developers shifted towards CSS for its ease of use and improved performance since CSS introduced @keyframes animations and transitions, offering a simpler, declarative way to create animations

With advancements in browser technology, CSS animations became more efficient, benefiting from browser optimizations while JavaScript frameworks like GSAP emerged, providing robust tools for complex animations, addressing some performance concerns

Choosing between CSS and JavaScript for animations today depends on project needs, with CSS preferred for straightforward animations due to its simplicity and performance advantages, while JavaScript is utilized for more intricate, interactive animations, making the best practice involve leveraging both technologies as appropriate to ensure optimal performance and maintainability

Your turn

1.Lights camera action

Create a presentation about CSS gradients, shadows, transitions, transformations, and animations

- Craft 15-30 slides detailing CSS features
- Include at least 8 code examples with your own original code
- Dedicate a section about box-shadow, gradient, and animation generators
- Provide references to sites with interesting CSS implementations
- Include your **html** and **css** files in the delivery

2.Stylish interactions

Create a web page to showcase various CSS effects

- Include a heading and a paragraph with some sample text
- Apply a linear gradient background to the heading
- Add a shadow to the paragraph text
- Use a transition to smoothly change the paragraph text color on hover
- Create a div and apply a transformation to rotate it 45 degrees
- Add an animation to the div that moves it from left to right continuously

Bonus: use radial gradient for the heading background and add a hover effect to pause the animation

References

CSS Box Shadow
w3schools CSS Shadow Effects
CSS Tricks box-shadow
CSS Tricks text-shadow

w3schools CSS Gradients
w3schools CSS Radial Gradients

Button with Gradient and Shadow example

References

Using CSS animations | MDN
CSS Animations

30 Creative and Unique CSS Animations