ORGANIZATION FOR DONATION:

Internet Archive

The Internet Archive's mission is universal access to all knowledge, where they are most know for their keyframes across web space time in the Wayback Machine.

https://archive.org

ARTICLE BLURB

Creating smooth transitions has been upgraded-- learn the latest tricks for entering and exiting content gracefully.

ANIMATING ENTRY EFFECTS - @STARTING-STYLE AND ALLOW-DISCRETE

Animating Entry Effects Demo

Animation is a critical component to interactive storytelling. An animation creates a journey, no matter how small, from point A to point B.

In classical animation, an artist will draw one frame at a time and a camera will take a snapshot of each moment of progression. For just one second of animation, 24 frames need to be drawn, captured, and sequenced.

For interactive content such as video games, web apps, and operating systems-- players, visitors, and customers expect there to be a stable 60 frames per second. Anything less than 60 individual images per second is perceptible to people with an eye tuned for it.

CSS ANIMATIONS

With CSS, there are two primary properties used for animation: transition and animation. Both properties allow customization based on how long the animation should last (duration), when it should start (delay), and how the in between frames should be "tweened" (easing).

Today, we're going to focus on a couple of aspects that can be utilized with the transition property in CSS: allow-discrete and @starting-style.

In the demo we'll build, we will gracefully fade-in all content as it appears in our web document.

Before we go further, we need to take a step back and zoom out.

We know animation is a journey from point A to point B, but what are the technical terms for point A and point B?

KEYFRAMES

A keyframe is the state of affairs at any moment in time. Given two states, we can calculate the difference between them and animate. This works great for things like numbers, but it doesn't work great for things like words.

For example, there are a myriad of ways to not show a piece of content with CSS. Two of the more popular CSS properties are using opacity (continuous) and display (discrete).

OPACITY

Opacity is continuous and takes a value between 0 and 1, inclusively. An opacity of .5 will be 50% transparent-- you'll be able to see content behind the content. An opacity of 0 is hidden and 1 is entirely visble.

If the opacity of point A is 0 and the opacity point B is 1, with an animation duration of 1000ms, the folowing statements are true.

At 0ms, opacity: 0;

At 250ms, opacity: .25;

At 500ms, opacity: .5;

At 900ms, opacity: .9;

At 1000ms, opacity: 1;

Writing each intermediate step out would be tedious in code, which is why the CSS engine tweens values on our behalf.

DISPLAY

Display values are discrete and may be defined as: inline, inline-block, flex, grid, inline-flex, inline-grid-- and all those properties will result in visible content.

Display may also be set to none, which will hide the content entirely.

Historically, CSS has not had a way to animate from a point A of display: block to a point B of display: none; or of a point A of display: none; to a point B of display: grid;

Animating between words, or discrete values, is hard. How can we do it?

ALLOW-DISCRETE

The transition property allows for a constituent property 'transition-behavior', which may be 'allow-discrete', which will allow CSS to animate between words.

In this example, we apply the transition-behavior of allow-discrete to all elements. The transition-properties we're using are visibility, transform, and opacity. Opacity and transform are both continuous properties that have transitioned since transition was implemented in CSS. Visibility is a discrete property and needs allow-discrete to transition properely.

MDN transition-behavior

CODE AND DEMO

@STARTING-STYLE

In the example, notice the use of @starting-style. This block is where we can define the initial keyframe we want to transition from, a new default value we as web authors define for our applications.

In the @starting-style block, we select all elements with the wildcard selector, and set the initial visibility to hidden with an opacity of 0.

When the page loads, these will revert to their browser defaults of visibility: visible and opacity: 1, creating a smooth transition.

MDN @starting-style

CONCLUSION

@starting-style and transition-behavior unlock to potentials for animated storytelling that previously required JavaScript to achieve similar effects. With these tools now baked into CSS directly, how will you use them?

While the example in this demo is intentionally light to stay focused on the basics of these properties, a great first project might be to build a modal that can gracefully fade-in and fade-out.

A common pitfall when implementing modals is for the hide animation to immediately hide with no animation, which generally happens when visibility: hidden; triggers, immediately when the modal should hide instead of at the end of the animation sequence.

By using transition-behavior: allow-discrete when hiding the modal, other CSS or JavaScript workarounds that have historically been used for a smooth effect when hiding are no longer necessary.

Let us know in the comments how you end up creating with these new toys!

- Ty from https://sillyz.computer