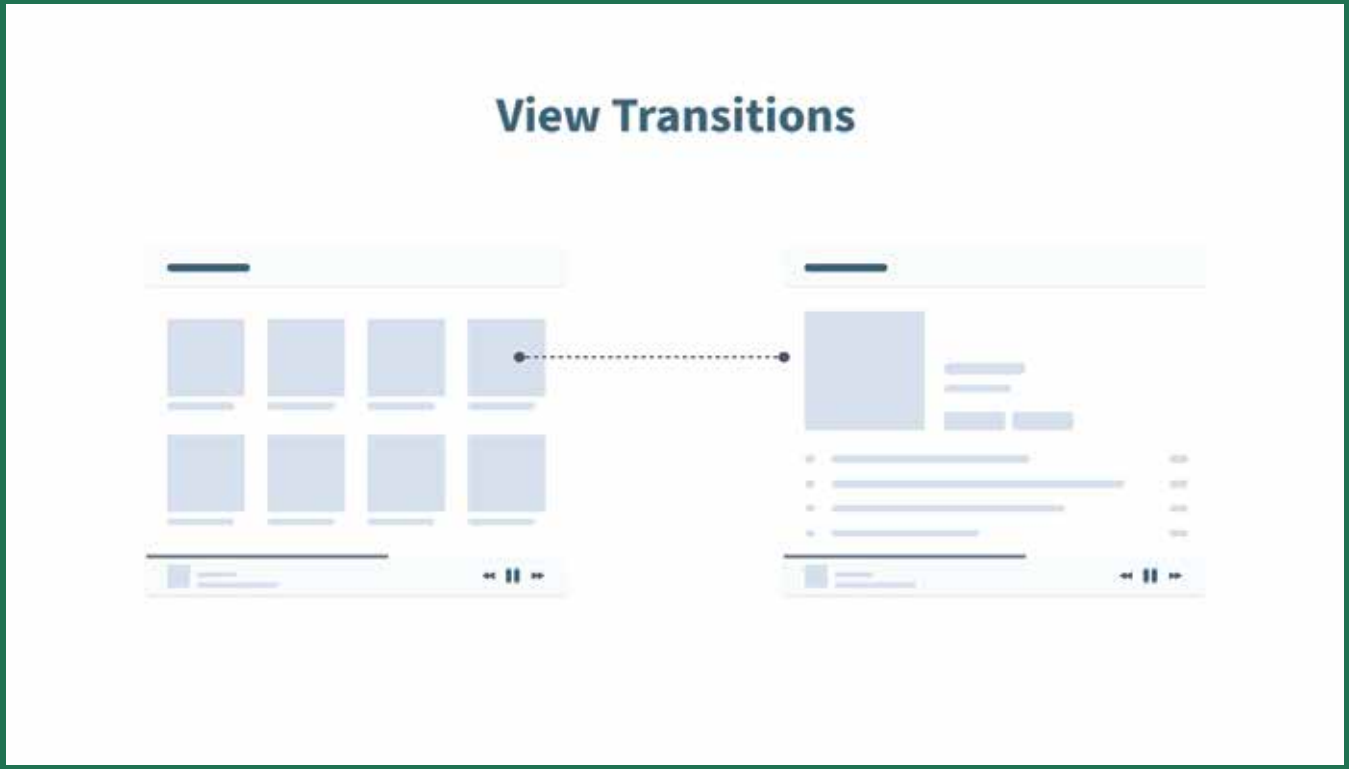
View Transition API

What are “view transitions”? How will they change the web? Can we use them now?

WHAT ARE VIEW TRANSITIONS?



The View Transition API provides a mechanism for easily creating animated transitions between different website views.

How It Works (The 4-Step Process):

Step 1: Capture

User triggers an action (clicks a button, navigates, filters a list)
Browser takes a snapshot of the current view state
All elements and their positions are recorded

Step 2: Update DOM

Your JavaScript code runs and updates the DOM
Content changes, elements are added/removed, layout shifts
All happens in a single atomic update

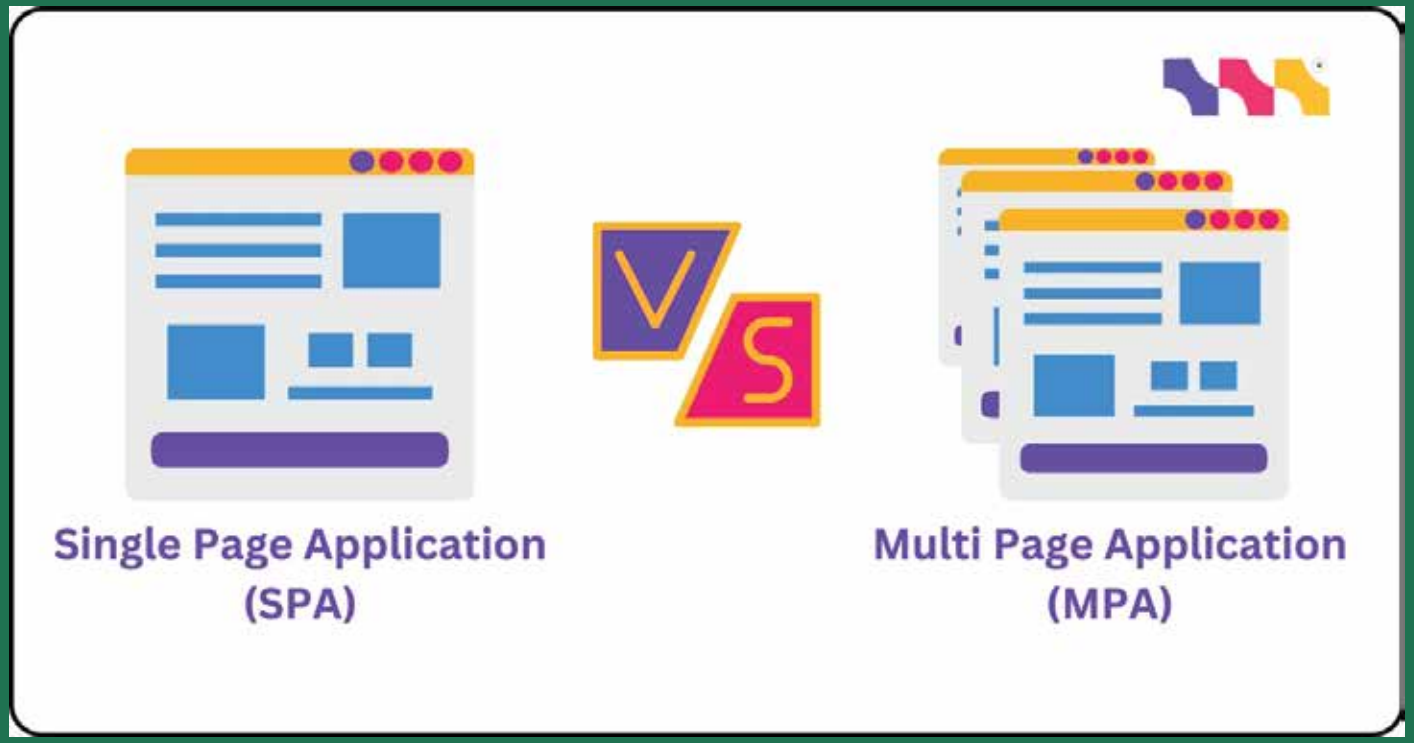
Step 3: Snapshot

Browser captures the new view state
It knows where every element was before and where it is now
It calculates position and size differences

Step 4: Animate

Browser creates CSS animations that morph between old and new states
Elements appear to move, scale, fade, transform smoothly
All at 60fps using GPU acceleration
Then the browser cleans up the temporary pseudo-elements

TWO TYPES OF VIEW TRANSITIONS:



Same-Document (SPA Transitions)

Used with document.startViewTransition()
For content changes within a single page
Examples: filtering, sorting, expanding sections, tab switching

Cross-Document (MPA Transitions)

Automatic on page navigation
Works between different pages
Both pages just need to opt-in with a CSS rule

Before View Transitions (Complex):

```javascript
// Without view transitions, you might do something like:
async function updateList(filter) {
  // Animate out old content
  await animateOut();

  // Update DOM
  filteredItems = items.filter(item => item.type === filter);
  renderList(filteredItems);

  // Animate in new content
  await animateIn();

  // Handle focus, accessibility, event listeners...
}
```

With View Transitions (Complex):

```javascript
function updateList(filter) {
  document.startViewTransition(() => {
    filteredItems = items.filter(item => item.type === filter);
    renderList(filteredItems);
  });
}
```

That's the entire difference. The browser handles all the animation logic.
Customizing the Animation (CSS):

```css
/* Default fade transition */
::view-transition-old(root) {
  animation: fadeOut 0.5s ease-out;
}

::view-transition-new(root) {
  animation: fadeIn 0.5s ease-in;
}

/* Or customize specific elements */
.product-image {
  view-transition-name: product-hero;
}

::view-transition-old(product-hero) {
  animation: zoomOut 0.4s cubic-bezier(0.4, 0, 1, 1);
}

::view-transition-new(product-hero) {
  animation: zoomIn 0.4s cubic-bezier(0, 0, 0.2, 1);
}
```

WHY IS THIS IMPORTANT?

Before View Transitions:

You needed to learn complicated animation libraries (GSAP, Framer Motion)
These libraries add extra code (50KB+) to your website
Different frameworks used different tools
Only professional developers could do it well
Websites often had no animations because it was too hard

After View Transitions:

Anyone can add smooth animations
Animations just work in the browser
No special libraries needed
Much less code to write
Even beginners can make professional-looking animations

HOW DOES THIS CHANGE THE WEB?

Change 1: Less Code to Write

Before: Write 50 lines of animation code
Now: Write 3 lines with View Transitions
You can actually focus on the real stuff instead of animation complexity.

Change 2: Websites Feel Better

Before: Many websites had no animations (too complicated)
Now: Anyone can add smooth animations
Regular websites will look as polished as big company websites.

Change 3: Works Everywhere

Before: React websites used one animation tool, Vue websites used another, vanilla JavaScript did something different
Now: Everyone uses View Transitions. Same tool for everyone.

Change 4: Web Apps Feel Like Phone Apps

Before: Web apps felt different from phone apps (less smooth)
Now: Web apps can be just as smooth as phone apps
Users won't feel the difference anymore.

CAN WE USE THEM NOW?

Desktop Browsers:

Chrome: 65%+ support
Firefox: 25%+ support (new!)
Safari: 10%+ support
Edge: (included in Chrome stats)

Mobile Browsers:

Chrome Mobile: 65%+ support
Firefox Mobile: 25%+ support (new!)
Safari iOS: 10%+ support (iOS 18+)
Samsung: 60%+ support

TOTAL: 90%+ of modern browsers support View Transitions

Current support is not widely adopted yet, as of the present moment, the support for the View Transition API is exclusively in Chrome version 111 and beyond.

LIMITATIONS

What View Transitions Are Good For:

- DOM state changes within a page
- Page navigations (same-origin)
- Filtering, sorting, expanding content
- Switching between tabs or sections
- Gallery/lightbox interactions

What They're NOT For:

- Streaming/real-time updates (they need discrete state changes)
- Complex 3D animations (use WebGL for that)
- Very large DOM changes (might impact performance)