Application Performance Recommendations

c/o ActiveVideo

Overview

All applications being deployed to CloudTV should be reviewed to identify any existing bottlenecks to performance on the CloudTV Platform. In general, identify high value optimizations first, focusing on (in order)

1. Reducing the scope of paint operations to the regions affected (no bleed over)
2. Reducing the number of paints
3. API considerations related to HTTP cache friendliness of

Paint Operation Scope

The reasons for #1 tend to fall into a typical set of solutions:

- set "overflow:hidden" on containers that hold elements that change or animate
- avoid triggering layouts when unnecessary (for example putting a border on something when focused, set the "box-sizing:border-box" to avoid a re-layout/paint).
- in console, enable "paint rects" to see where paints are happening

Reducing number of paints

The reasons for #2 tend to fall into a typical set of solutions as well

- animate using linear transformations (i.e. transform/webkit-transform and translate3d, scaleX, scaleY, rotate)-
- animate in a deterministic manner via CSS transitions and not via setTimeout as that approach lead to greedy animations that are unaware of any encoding cost post-paint (i.e. the MPEG encoding budget available must be spread over the whole animation, not just the 1st couple frames).

MPEG Encoded Animations:

- By specifying the start/end of an animation, the animation sequence can be throttled for peak performance over the time of the animation
- Animating frame-by-frame over properties that force layouts like `top`, `left`, `width`, `border-width` will generally cost more to paint, than their layer counterparts.
- By using CSS transitions, CloudTV can introduce optimizations that more efficiently re-use pixel data during the life of the animation.

API Cache Friendliness

The desired architecture for APIs is designed to promote reusability of HTTP objects across a footprint.

1. Decouple metadata and user data - API calls for metadata (guide, VOD, cast and crew, etc) generally contain the same data for a large footprint of users, and can be served from cache 99% of the time if they are not filled with any user/ subscriber specific information.

API Cache Friendliness Examples

For example, VOD catalogues can be broken into 2 API calls

1. VOD Tree (shared by all users within a headend, so /vod?headend=someHeadend)
2. My Rentals (personalized for a user) - this API call should be asynchronous and not block UI from displaying the VOD Tree as soon as it is returned