JS API for Media Controls and Media Focus

When: Sunday 11/17/13 3 -3 pm, 18 people

Moderator: Boris Smus, Google

Boris presents slides with problem statement and proposal for how to handle media controls in the browser. Summary:

Many ways of controlling media:

- Media keys
- Headset buttons
- Gestures control
- Voice control
- Remote controls
- Keyboard shortcuts
- ...

No way to handle this in browsers today

Partially solved for Android and iOS

Proposed Media Focus model:

- Foreground tab if it accepts media controls
- Otherwise, route to tab that was most recently controlled
- Keep a focus stack

Prototype to illustrate proposal: http://borismus.github.io/media-control-prototype/

What should happen when one player is playing but you bring another into focus?

Last thing that was interacted with or last playing? Boris: believe the last actively controlled

A button to kill everything suggested

- Suggested that always stop everything when pressing pause/stop
- What happens if you're in a VC? You don't want to stop that just everything else playing

Proposal to have a separate API for media controls (not a generic controls API)

How do we know that the user interacted and not e.g., and ad?

- Similar to mobile non-support of autoplay
- Spec should tell how the browser reacts to the controls

Going forward

1. Agree this makes sense
2. Implement extension to test it out?
3. Implement spec

Co-authors for spec?

Is this useful?

- No-one explicitely negative

Discussion

Ami: Does this exist at the OS level?

- Yes on mobile, not as clear for desktop
- Nasty hacks from e.g. Spotify to get access to media keys on mac

First maybe just to get media keys to control foreground tab

- Suggested to file an Apple Radar bug (Eric Carlson: black hole but it works)

Delineation between long playing versus short snippets (e.g. games)

- Mobile OSs support the concept of audio ducking

Kill all when hitting stop/pause does not sound popular

Concerns that it will be hard to do because not all sounds are in the same category. In Android and iOS the category is specified when opening a sound source

You don’t have to subscribe to media events if you don’t want the controls to take effect

Important that this is something a web app can take advantage of not a forced situation

What happens on the mobile systems where ducking is happening?

Distinguish between app level and browser level event controls. Only if the events get to the browser

Only user activated events

We can’t assume that the page has control as the OS level state can impact that

Does the browser just stop based on a stop event or does it just tell the app and it has to stop?

When one stops playing do you want to switch focus? Pages don’t know

Concerns:

- Should any page be able to request focus? E.g. ads

Request focus probably happens in the beginning and everytime the user interact with it

Can we prevent background pages to take focus? Problem for e.g. pages loading in the background

What happens when you load a lot of pages without starting to play? They get focus in the reverse order of how they were opened

What part of the behavior belongs to the spec and what to the implementation?

- Enough to spec the behavior for media keys and then all other controls can be mapped to them
- Browser could implement stop all
- Audio indicator (just launched in Chrome) helps a lot

Major discussion around whether stop should stop all

Proposal:

- Don't implement requestMediaFocus()
- Everytime you want to get focus requires active play event
- Browser knows if play/stop was because of user activity or not
- What about when timers are set?

Wrap up

General consensus that this is a good idea! Several point out that they use e.g. Rdio app rather than browser page just because they can't control with media keys in the browser

What's next?

- Get some more feedback and co-authors for the spec proposal
- Preferable to join a WG. Consider Indie UI
- Extensions approach interesting
- Proposal for a mediator (coupled to the window) instead of window.addEventListener
    - Developers don't really want to tie everything to windows
    -