Second Screen for the Web

Breakout Session TPAC 2013

Dominik Röttsches, Intel OTC Finland, @abrax5
Hongbo Min, Intel OTC PRC
Anssi Kostiainen, Intel OTC Finland, @anssik

11/13/2013

irc://irc.w3.org:6665/#webscreens

This Breakout Session Is Intended To...

- Introduce our idea for using secondary displays from the web in the form of a presentation and demo
- Invite and encourage interested attendees, especially implementors, to join the newly founded Second Screen Presentation Community Group that covers the topic
- If time permits, have a small round of Q&A and discussion

Today

- Introduction „Second Display for the Web“
- Demos
- (Remote Displays Technology Overview)
- Presentation API Proposal
- Current Status & Getting Involved
- Q&A, Discussion

Introduction

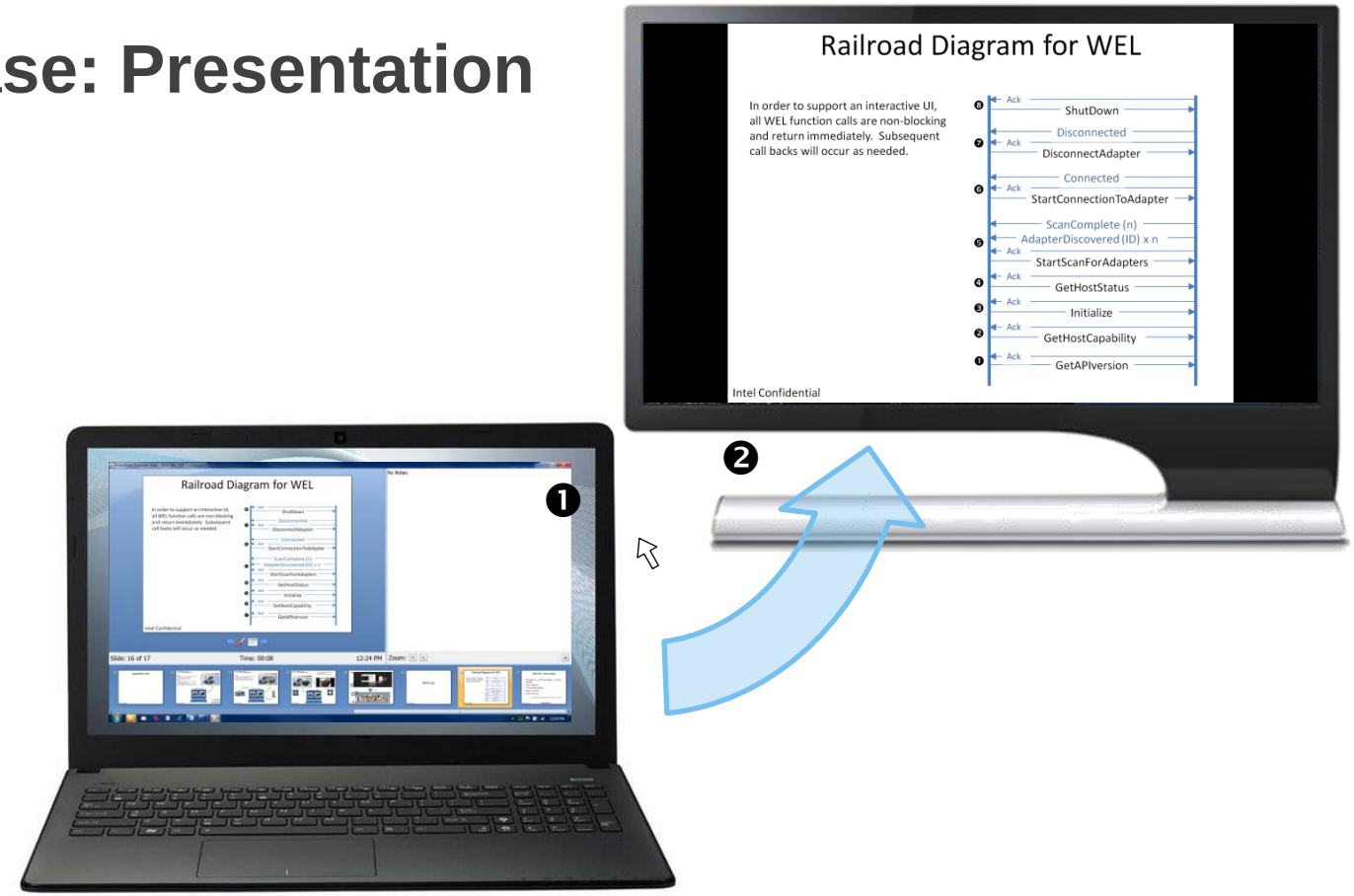
Second Display for the Web?

Enabling the User Agent to show web content full screen on nearby displays, such as TVs, projectors etc.

“Second Display” Clarification

- Second Display in this context means presenting on nearby displays from a UA on your primary device (laptop, handheld) to the display
- Different compared to Web & TV context, where "Second Screen" is often used to mean an application on a tablet, phone or other device associated with a program simultaneously broadcast on a TV.

Use Case: Presentation



(Slide material courtesy of Steve Barile, PCCG)

Use Case: Video Sharing

Use Case: Gaming

Demo

Remote Displays Technology Overview

Remote Display Technologies

- Miracast & Intel Wireless Display
  - On Windows 8 & 8.1
  - On Android
- Airplay
  - On iOS & OS X – multi-monitor support in OS X Mavericks
- DLNA-based
  - Xbox, Windows Media, PS3, TVs
- Chromecast
  - Partially web-based, two user agents connecting

OS APIs for Second Displays

- Desktop OS
  - “Invisible Monitor cable”
  - Wireless displays are transparent to applications
  - API-wise appear just like any other monitor, Display Enumeration & Info
  - Allows Mirror/Clone Mode, Extended Mode
- Mobile OS
  - Special APIs to access secondary displays
  - Mirroring and second-screen mode
  - Compatible with Widget/Views architecture of the OS

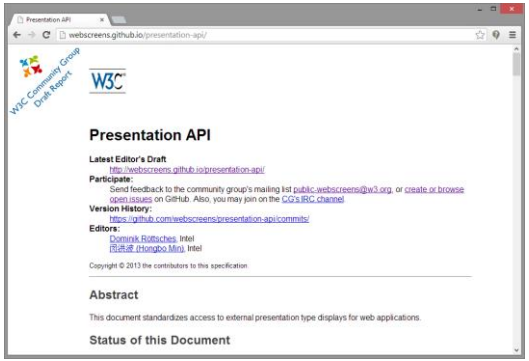
Web

- Popup Blocking (using secondary windows is frowned upon)
- Fullscreen API established
- Network Service Discovery API (covering DLNA, Zeroconf, DIAL)
- Alternative Sinks for Video Elements
  - Microsoft’s PlayTo extensions for the video element
  - Apple’s Airplay support for media elements
  - Sink enumeration for Video and Audio Elements proposal http://lists.w3.org/Archives/Public/public-html/2013Aug/0101.html
- Chromecast
  - Integration in Android, Chrome browser

Enabling Second Display Usage for the Web: Presentation API

Presentation API Proposal

http://webscreens.github.io/presentation-api/



- Abstracts away physical display connection technologies
- Simple, unobtrusive API which enables a wide variety of use cases
- A bit like Fullscreen API for a nearby TV

API Preview

IDL

```
partial interface Navigator {
   readonly attribute Presentation presentation;
}

interface Presentation :  EventTarget {
  Promise requestShow(optional DOMString url = "about:blank");

  readonly attribute boolean displayAvailable;
  attribute EventHandler ondisplayavailablechange;
};
```

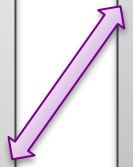

Presentation API Example Video Case

Phone / Laptop

```
<script>
var p = navigator.presentation.requestShow(
  “video.html”);

p.then(showSuccess, showError);

function showSuccess(win) {
win.postMessage('http://example.org/video.mp4',
'/');
}
</script>
```



TV / Second Screen

```
<video>

<script>
var v =
document.querySelector('video');
window.onmessage = function (event) {
  v.src = event.data;
  v.play();
};
</script>
```


Second Screen Authoring

- Responsive Design for adjusting Layout to screen size
- User Interaction remains on the primary display, avoid user interaction on the second display (no prompts, alerts, form entry, etc.)

Presentation API Key Features

- Presentation API
  - *requestShow* will create a presentation window on the secondary display for the given URL
  - Returns *WindowProxy* object on success
  - Presentation window always frameless, full screen window – common denominator between mobile and desktop scenarios
  - *displayAvailable* flag indicates if there is at least one available secondary display
  - Web Messaging between opener and second screen window
  - Promise based API
- Security: Fingerprinting risk mitigated by not leaking a display enumeration footprint

Open Issue: Single User Agent vs. Distributed UA

- Currently, Presentation API assumes one UA preparing the rendering for multiple screens
  - requestShow() call returns WindowProxy
- Alternatively: Modify specification to allow distributed UAs – primary UA on device, secondary UA on TV
  - Implement discovery between UAs
  - Return a MessagePort from requestShow() call, no direct remote DOM access

Current Status & Getting Involved

Community Group

- Community Group was just formed thanks to several supporters http://www.w3.org/community/webscreens/
  - From Browser Vendor side: Mozilla joined and investigates implementation
- You're invited to participate!
  - Additional use-cases?
  - Feedback on the Presentation API proposal?
  - Your new proposal?
- Currently Acting Chair

Presentation API Draft

- Presentation API draft published on Github pages http://webscreens.github.io/presentation-api/
  - Our input to the group, open for discussion
  - Ready for proof of concept implementations
- Proof of Concept implementations ready
  - Based on Chromium Tip of Tree
  - Based on Intel's open source Crosswalk runtime (https://crosswalk-project.org/)

Participate

Second Screen Presentation Community Group
http://www.w3.org/community/webscreens/

- Review the API proposal and charter
- Join the mailing list http://lists.w3.org/Archives/Public/public-webscreens/
- Discuss on #webscreens W3C IRC
- Find more information on the Wiki
  http://www.w3.org/community/webscreens/wiki/Main_Page

Q&A, Discussion

http://www.w3.org/wiki/TPAC2013/session-second-screen-web