Web Intents Demo

- Slides represented as the Slidy microformat
- Android app on phone to act as a remote controller
- The remote controller app advertizes itself via UPnP
- Browser listens for UPnP devices in background
- Slide requests intent for remote controller
- User asked for permission to use this device
- Wrist gesture on phone signals prev/next slide

start gestures

Gestures

- App listens for sensor change events on gyroscope X axis
- Heuristic approach to recognizing wrist rotation movements
- Clients gets gesture events via Server-Sent events or HTTP long polling
- Web sockets would be better, but I didn't have time to implement it as part of a native Android app
- Not sure how to express this as a UPnP service definition file

Web Intents Demo

Using Greg, James and Paul's proposal and a leap of faith, I would have:

```
// the intent name is just for example
var intent = new Intent("http://example.com/remote-controller");
window.navigator.startActivity(intent, function(data)
{
  var channel = new MessageChannel();
  data.service.postMessage('hello',
                           'http://example.com',
                           [channel.port2]);
  channel.port1.onmessage = function (event)
  {
    if (event.data.indexOf("next") >= 0)
      next_slide();
    else if (event.data.indexOf("prev") >= 0)
      prev_slide();
  };
});
```

I'm not quite sure about the postMessage step, or whether Greg at al. even accept the possibility of persistent services with web intents.

Other ideas for passing the port: as an argument to new Intent() or to startActivity().

Web Intents Demo

Using Ian Hickson's proposal I would have:

```
// the intent name is just for example
var port = navigator.handleIntent("remote-controller");

port.onmessage = function (event)
{
  if (event.data.indexOf("next") >= 0)
    next_slide();
  else if (event.data.indexOf("prev") >= 0)
    prev_slide();
};
```

That's very nice!

div class="slide">

Using Ian Hickson's proposal for simple cases

Share intent with some data and no response:

```
navigator.handleIntent('share').postMessage(data);
```

The service page would handle the share intent with:

```
<body onintent="share(event.data)">
```

having registered itself as an intent handler with:

```
navigator.registerIntentHandler('share', 'text/uri-list');
```

That's very easy to understand.

Intent registration

Greg et al. propose new element in the document head or body:

```
<intent action="http://webintents.org/edit"
 type="text/uri-list;type=image/*,image/*"></intent>
```

- Browser developers use complex rules of thumb for dealing with head markup
- Have we checked with browser developers as to their opinion on adding the intent element?
  - *Ian Hickson, HTML5 editor in chief, advised against intent element*
  - *What about Web Kit, Firefox, Internet Explorer, Opera, etc. ??*
- What about web developers, will they understand the implications?
- What's wrong with a registration API called from the web page script?
- Use microformat or meta element for indexing by searching engine

User friendly names for services

- The intent name describes a *service type*
- There may several reasonable alternative services for given type
- I may have networked screens in several rooms in my house
- At work, there may be screens in each meeting room
- How do we give human meaningful names for each service?
- And share such names when a service is accessible to multiple people?

Asking the stakeholders

Why not seek broader feedback on design choices from:

- Browser developers
- Regular Web developers
- Library developers
- App developer ecosystems, e.g. Facebook
- People who develop home networking solutions

Implementing Web Intents

Somehow we need add handleIntent() to navigator, along with the logic for specific services

What techniques could we use to implement Web Intents?

- Web page library script, but ...
- Browser extension, e.g. chrome extension
- Native code extension to the browser
- Some combination of the above?

Using Chrome extensions for Web Intents

1. Background page script to manage intents
2. Injected content script running in its own sandbox, which injects script element in page DOM to load ...
3. Injected page script running in page's sandbox that adds navigator.handleIntent

This is crazily complicated ...

- Requires dummy elements in DOM to pass data between injected page script and content script using DOM events for signalling
- Requires message channel to pass events/data between content script and the extension's background script
- Further logic to implement a specific service
- Argh... let me out of this madhouse!

Native implementation of Web Intents

- Obvious next step!
- Getting my toes wet with Web Kit source code.
- Will no doubt raise all kinds of interesting issues, e.g. relating to security context, especially for inline service page disposition.
- Challenges for creating generic UPnP solution based upon service declarations
- Probable need for extensions to UPnP declaration format, e.g. to describe use of websockets

Web Intents Demo

A networked device can offer multiple services, e.g.

vibrate

photo