Web App Vs Web Site

Some tricks of the trade

Laurent Hasson
@ldhasson, lhasson@rim.com
Technical Director, BlackBerry Web Platform


INTRODUCTION

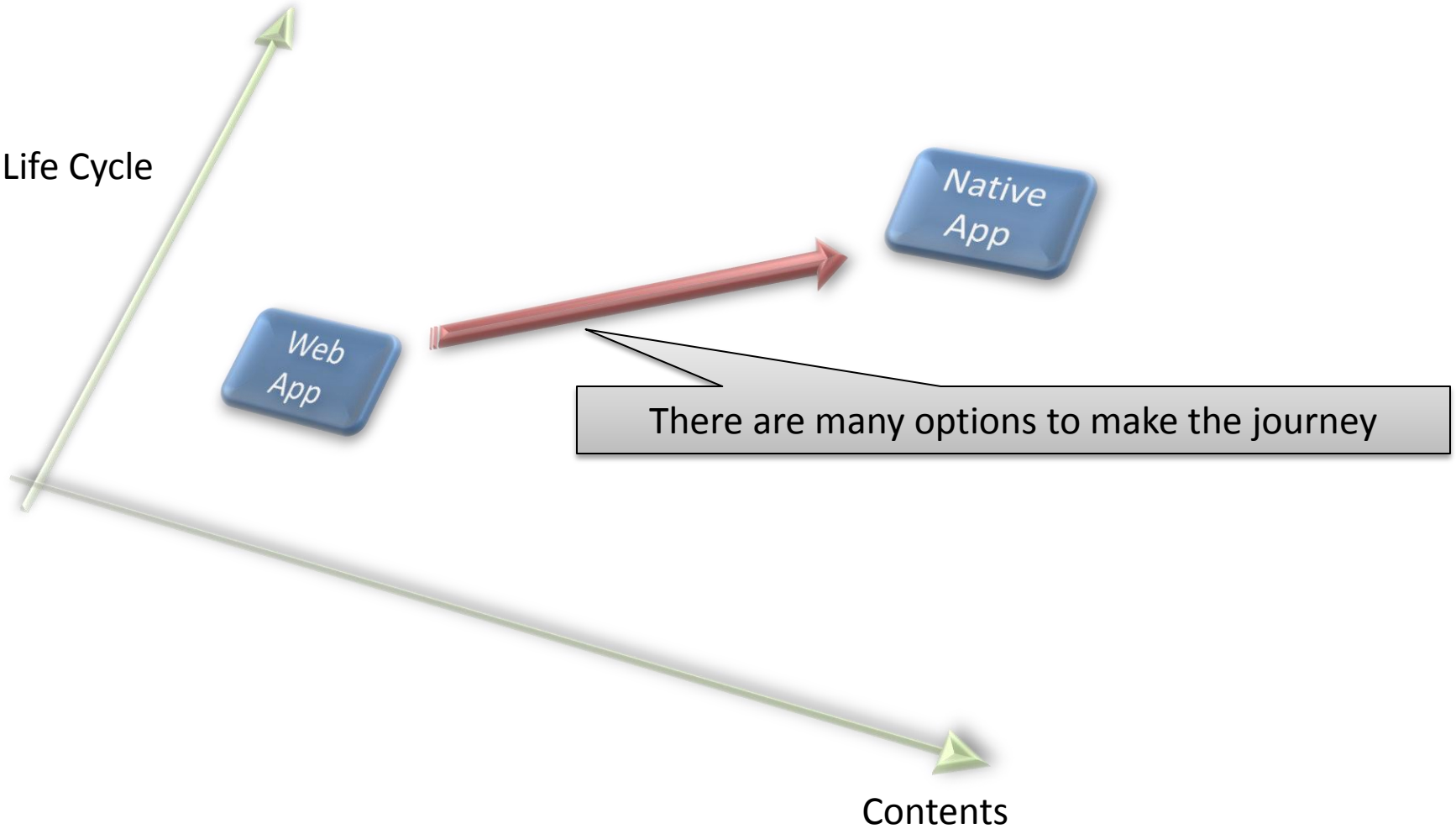
Web App Or Web Site?

- Lots of controversies and differing opinions from very smart people on the Web
  - Should a mobile Web app be more Weby, or more Appy?

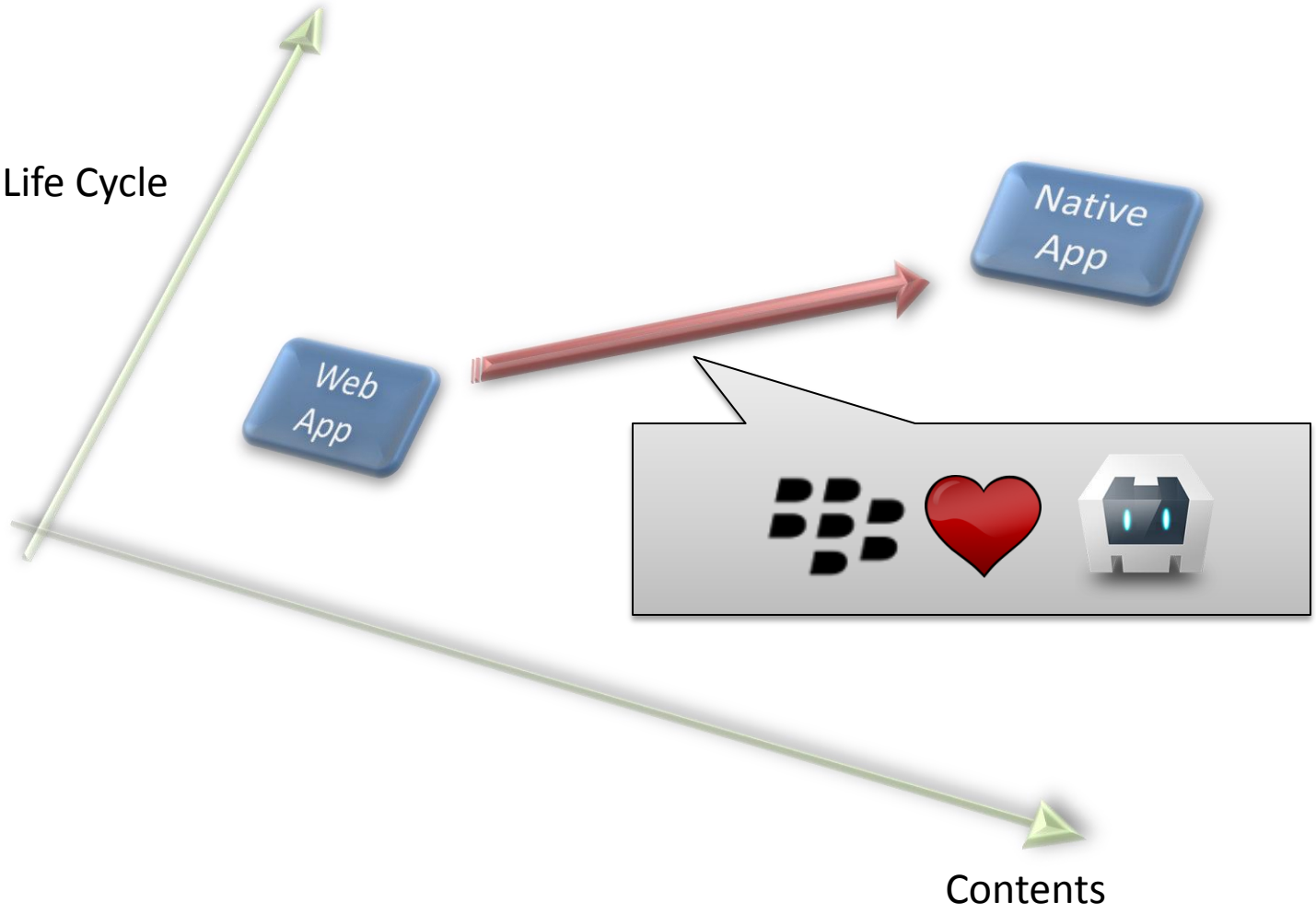
- My take: users today are conditioned to the App Life Cycle
  - Download something
  - Install something
  - Have an icon on the home screen
  - The App takes the entire screen real estate
  - The App is integrated with the device

- But users really don't care how the app was built!!!
  - So use Web technologies for the job!!

Two Dimensions to App Experience



Two Dimensions to App Experience



Say no to NIBS

* **N**ative **I**s **B**etter **S**yndrome

- The “Native Is Better” crowd are missing the point of the Web
  - It’s the scale of the market stupid!
  - It’s powerful, cross-platform, and an abundant skill set.
- This is not to say that Web is better than Native
  - That would be silly
- But the Web is absolutely competitive
  - Most types of apps can now be built very nicely using Web technologies
  - The gap is narrow today, keeps on getting narrower, and fast.
- Make no mistake
  - Native is the competition to Web

Web App Design Goals

- Single-page app metaphor
  - DOM manipulation for panels
  - And if not, mask page loading transitions
- Lack of browser chrome
  - Links, Bookmarks and Back
- Browser Events
  - Targeting multiple platforms means managing touches and clicks equally
- Viewport Management
  - Horizontal bounciness is bad
  - Zooming can be dangerous


VIEWPORT

When a pixel is no longer a pixel

- The desktop viewport is WYSIWYG
  - Window dimensions are well known
- On mobile, it's a whole other story
  - What you see is a viewport on a larger "thing"
    - Panning is a fact of life for most users
    - Zooming is a critical part of the browser UX
- And pixels are no longer display-relevant
  - High DPI and other display technologies have changed the meaning of what a pixel is
  - Better res without breaking the Web

CSS Pixels

- These are the pixels we deal with everyday
  - CSS definitions
  - Inline dimensions in HTML
  - Media queries
- On desktop, a CSS px == 1 device px
- On mobile, don't even try
  - The device vendor decides what the mapping is
  - You typically can't tell programmatically what the ratio is
  - So deal with those abstract pixels
    - On BlackBerry, default is 160DPI


HUH?

How many Pixel?

- screen.width/height
  - Device pixels
  - Entire screen, pretty useless as it's really a desktop concept
- document.documentElement.clientWidth/clientHeight
  - CSS pixels: 768px (usable space, minus browser chrome, so height is different)
  - Useful for media queries
- window.innerWidth/innerHeight
  - CSS Pixels of viewport: Depending on zoom level
  - DPI-scaled: 350px width (native res scaled to 160dpi for zoom-level of 1)
- window.pageXOffset/pageYOffset
  - CSS Pixels of scrolling

Viewport and Media Queries

```
<meta name="viewport"
      content="height=device-height, width=device-width,
               initial-scale=1.0, maximum-scale=1.0,
               user-scalable=0">
```

- Makes sure your page doesn't flow beyond screen dimensions (including margins and borders)
- Works as long as your content doesn't force overflow
- Deal in %

- @media all and (max-width: 480px) { … }
  - There are other flexible syntaxes for CSS media queries

Orientation: Landscape or Portrait

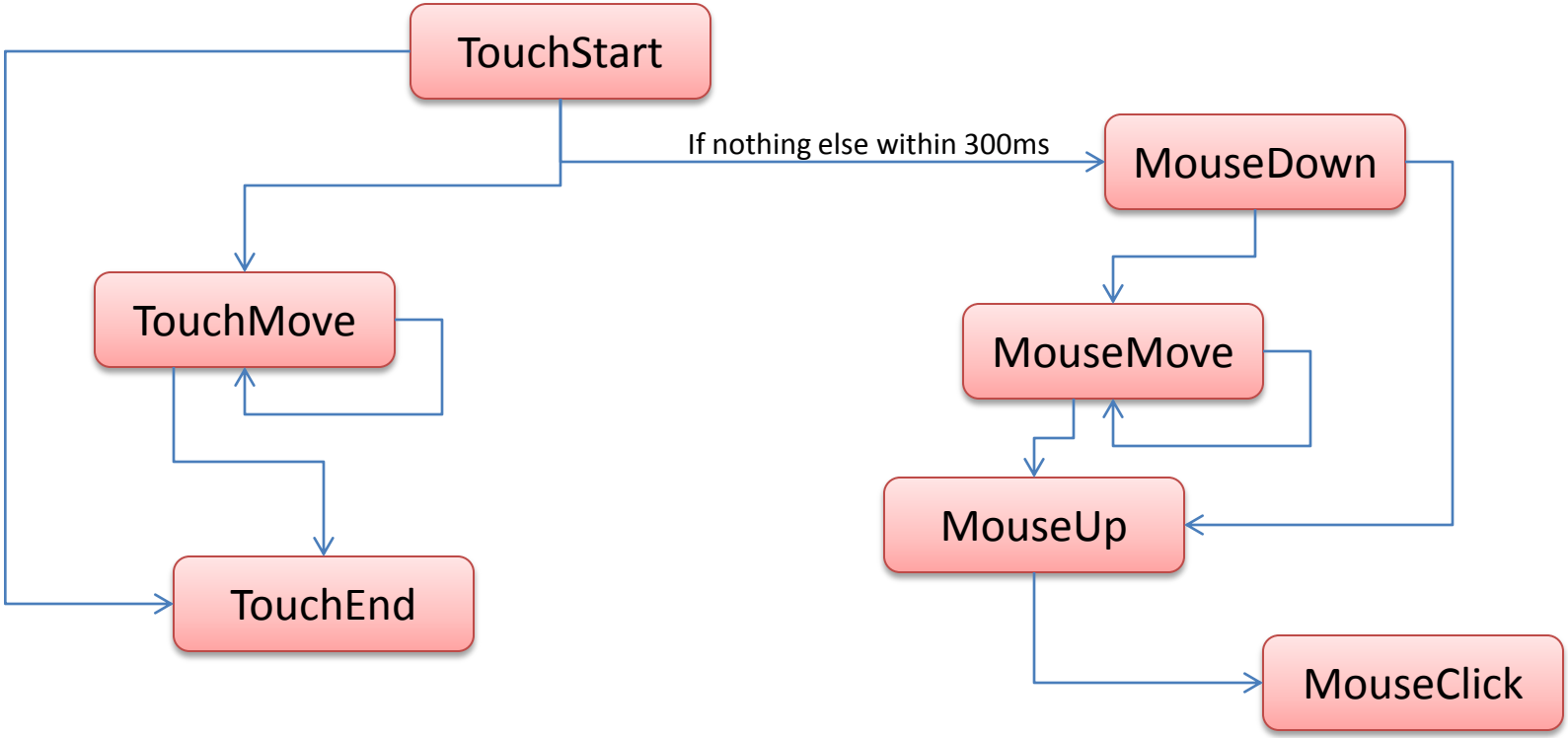
- Make sure you test in both modes
- Or lock the orientation
  - There are configuration flags available through packagers such as Apache Cordova
- Upcoming ScreenOrientation API
  - screen.orientation="portrait"|"landscape"|…, screen.lockOrientation("…"), screen.unlockOrientation(), screen.onOrientationChange
  - Languishing…


EVENTS: TOUCHES, CLICKS…

World Wide Web == Wild Wild West

- Some platforms do touch well
- Others simulate clicks
  - With delays and quirks
- Behaviors expected when preventDefault() are not consistent
- Plus, you may want to create Web content that also works in a browser
  - That's still an important goal if it makes sense to you

Your mileage may vary



- Delay is not the same everywhere
- The interplay between touch and mouse events vary
- The effect of onPreventDetault() vary

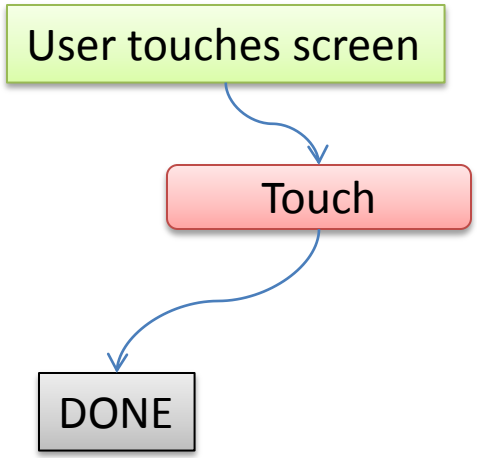
The Infamous Delay

- When the browser gets a touch, it needs to wait to see if there is a move or not
  - Some of the browser's own UX for pan/zoom need that
  - Typically 300ms
- In your app, you should take control of that
  - You typically don't want zooming gestures
  - You know what you expect from your users
  - You don't want that 300ms delay

touch-event-mode: BlackBerry Only ☹

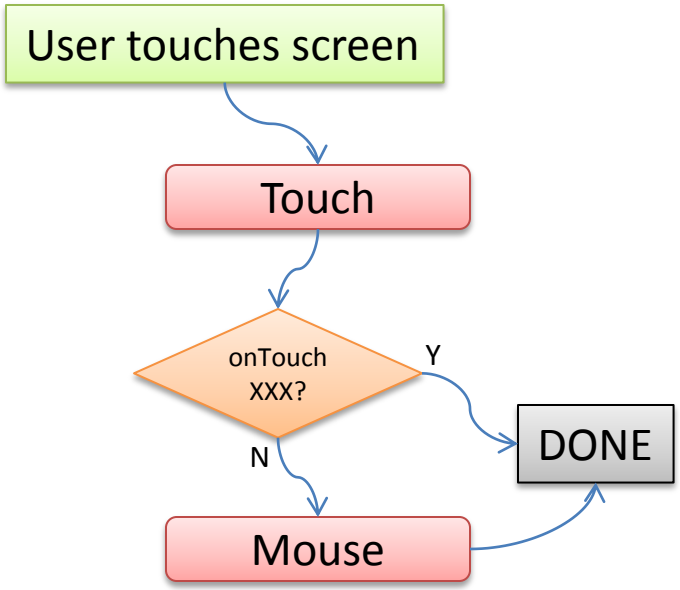
```
<meta name="touch-event-
   mode" content="native">
```

- WebPage gets TouchEvents only from touch screen.



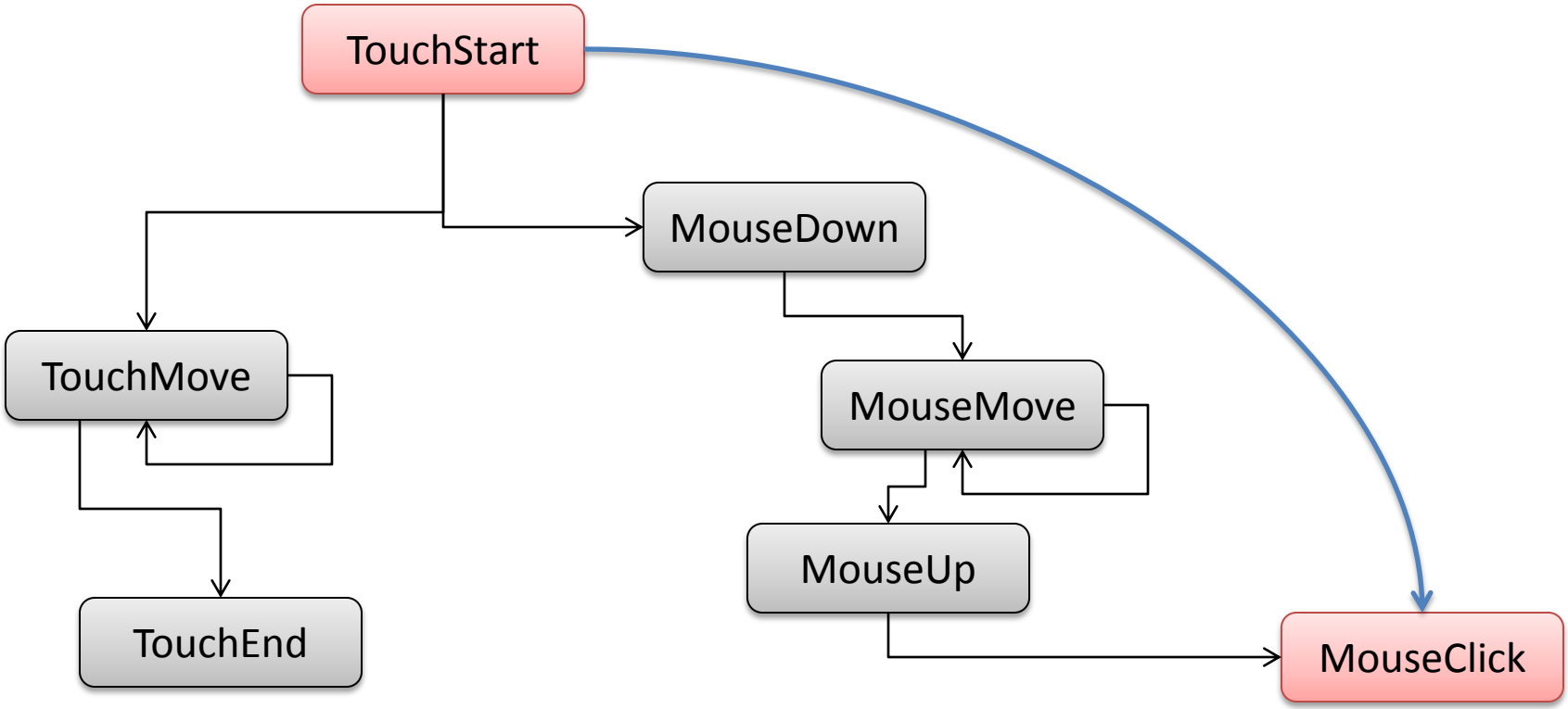

```
<meta name="touch-event-
   mode" content="pure-with-
   mouse-conversion">
```

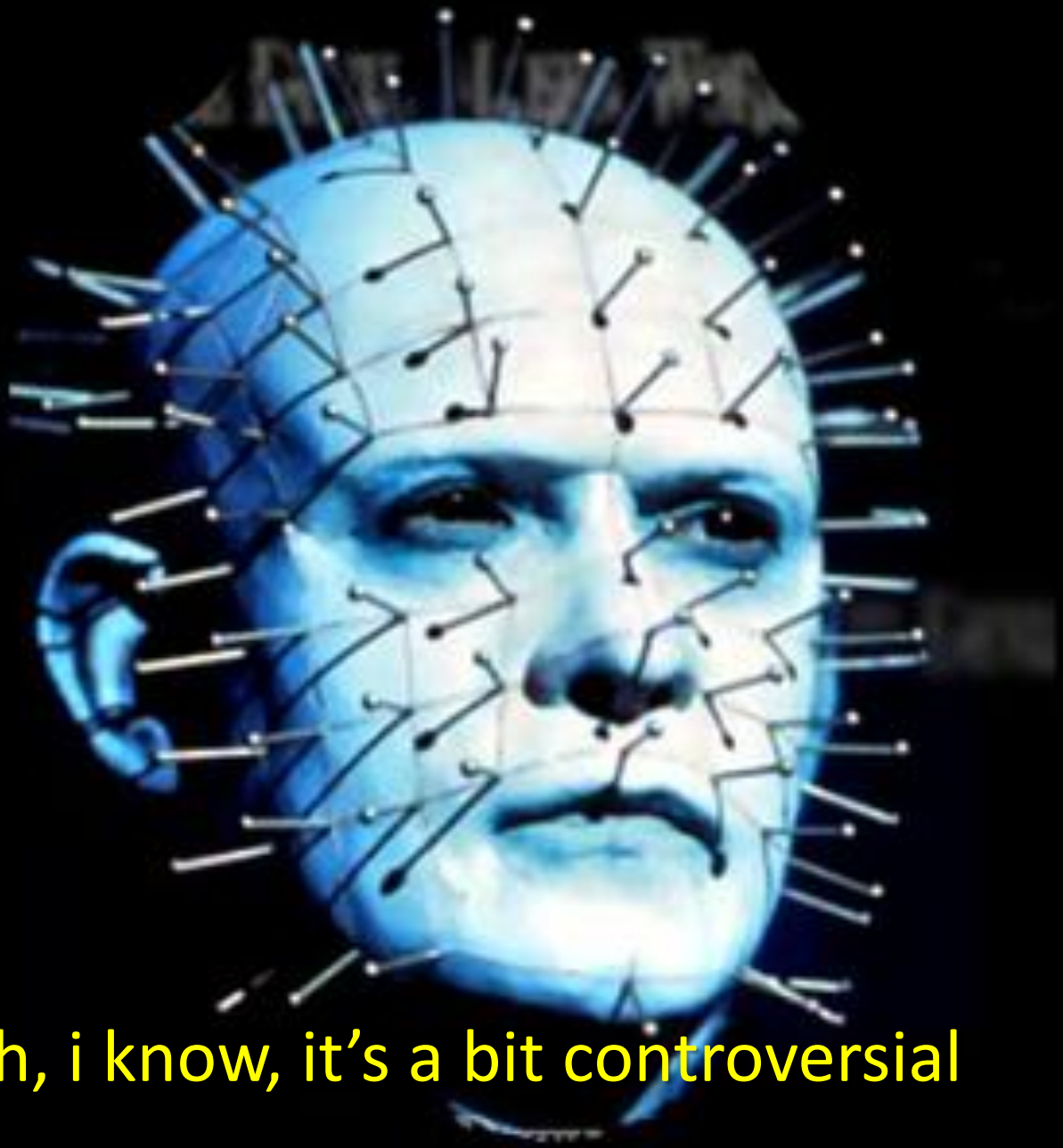
- WebPage gets both Touch events and Mouse events.



Browser UX disabled: DoubleTap zoom, pinch zoom, text selection and context menus

Control Your Destiny





Yeah, i know, it’s a bit controversial

Custom Event Handling

```
if (Touch)
 {
   document.ontouchend = function(event)
    {
      event.preventDefault();
      var e = event.target;
      while (e && !e.click)
       e = e.parentNode;
      if (!e || !e.click)
       return;
      event.clientX = event.changedTouches[0].pageX;
      event.clientY = event.changedTouches[0].pageY;
      e.click(event);
    }
 }
```

Blocks Browser UX behaviors

You may have to walk up the parent nodes to find an actual click() handler.

Convert touch coordinates to click coordinates

Invoke the onClick handler

You may want to do things differently to suit your app requirements, but the core idea is here

Other virtual events systems

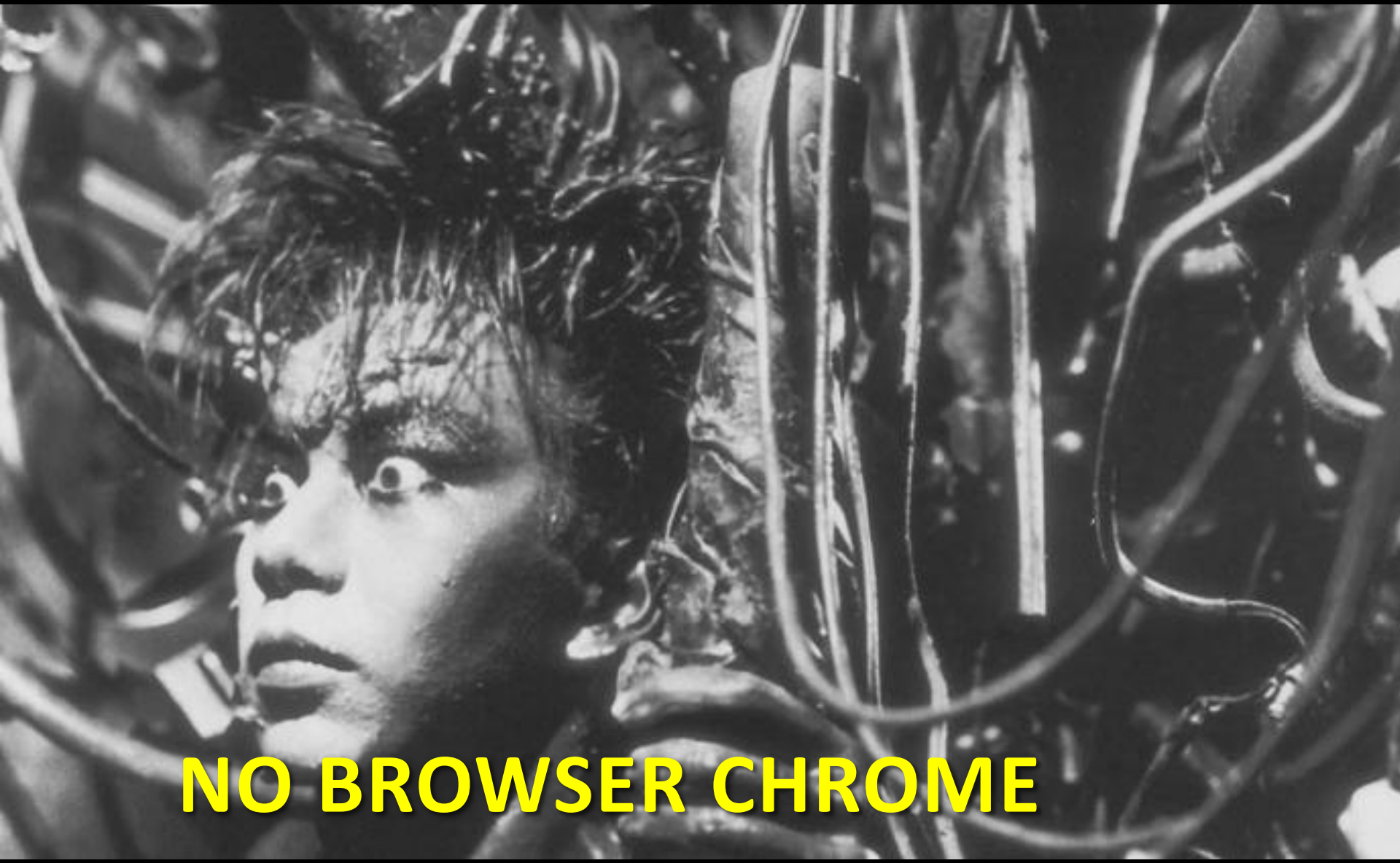
- Maps deviceOrientation events to key events
  - Simulates up/down/left/right arrows, or WASD or whatever
  - Most virtual keyboards don't have arrow keys
  - Most virtual keyboards wouldn't make sense in games anyways
  - Redirects to onKeyUp
- Famous frameworks, such as jQueryMobile, do it too, to gain control of, and unify, events.

user-select

- There is one more browser UX you may want to disable as well
  - User selection can be very annoying when in an app

```
body {
    user-select: none;
 }
```

- Or do that on selected sub-DOMs of your app
- In some cases, it makes sense, and you want to reuse the browser mechanisms for that


NO BROWSER CHROME

Arguably, the Web is about Navigation

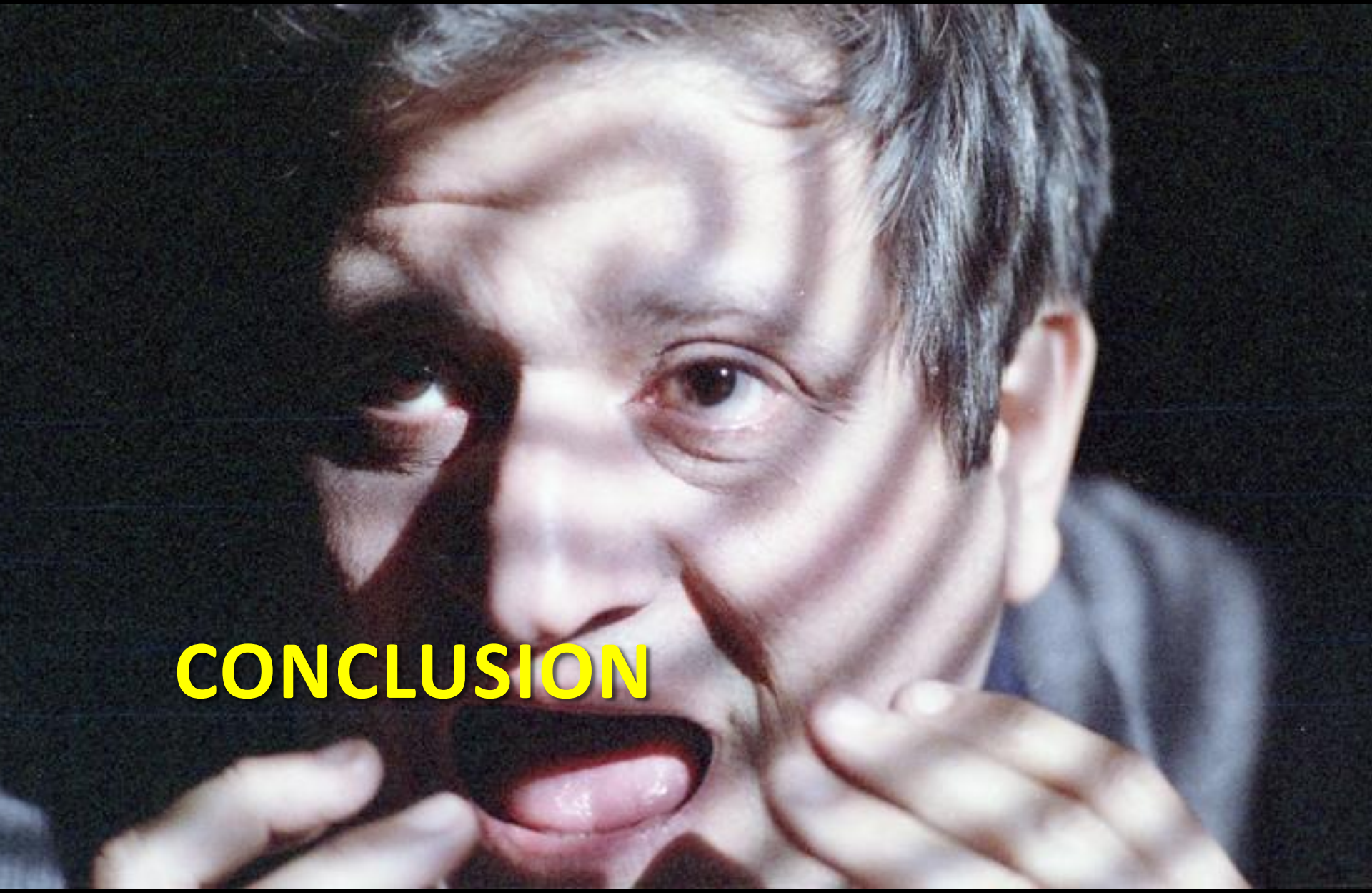
- You don't realize how much the Browser's skin does with your site until you don't have it anymore
- So many sites can still paint you into a corner where you can't escape without the back button
  - Even some big sites do it (I won't name names!)
- How about navigation
  - Many APIs exist now, and are well supported to help you.

Bookmarks == saved application states

- Why wouldn't you let your users create in-app bookmarks for content they specially like?
- Doing an App now doesn't mean you have to forget about REST principles and what has made the Web so cool!
- You need
  - The History API
  - Local storage
  - Some UI to let your users manage their "bookmarks"

And for debugging?

location.reload(true)


CONCLUSION

Web Apps <> Web Sites

- If you subscribe to the idea that Web apps must be Appy, then do things a bit differently
  - Viewport management must be explicit
  - Manage your own events
  - Navigation is different due to lack of chrome
- But don't drop what makes the web so cool
  - Navigation and Bookmarks are powerful features
- And debugging is hard, unless you do it on a BlackBerry
  - Best mobile browser AND built-in full remote WebInspector

The End – Thank You

References

Slide 2: The Matrix (1999)

Slide 5: BlackBerry Loves Apache Cordova

Slide 8: Poltergeist (1982)

Slide 11: Videodrome (1983)

Slide 15: Nightmare On Elm Street (1984)

Slide 21: Hellraiser (1987)

Slide 25: Tetsuo (1989)

Slide 29: Anguish (1987)

Slide 31: Masters Of Horror: Family (2006)