Developing for touch

Peter-Paul Koch
http://quirksmode.org
http://twitter.com/ppk
Mobilism, 16 and 17 May 2013
Examples

• Please open the following page on your phone:

• [http://quirksmode.org/touchevents](http://quirksmode.org/touchevents)

• It gives links to the test files I’ll refer to later

• If you have a Windows 8 device, be nice and show your neighbours what’s going on.
Best Viewed With

Your site should be best viewed on any device

- Desktop (all five browsers)
- Phone (any, not just iPhone and Android)
- Tablet (any, not just iPad)
- TV
- Game consoles
- Cars
- Refrigerators
Best Viewed With

Primary interaction modes:

<table>
<thead>
<tr>
<th></th>
<th>Mouse</th>
<th>Keyboard</th>
<th>Touch</th>
<th>Voice</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Desktop/laptop</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tablet</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TV</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Car</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Refrigerator</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Consoles</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓ ✓ ✓</td>
<td>✓ ✓ ✓</td>
</tr>
</tbody>
</table>

Take with large grain of salt. Interaction modes will converge - but haven’t quite yet.
It’s all about events
Stick with click
Stick with click

- click is not a mouse event
- click really means “activate”
- “I am now going to use this element for the task it’s meant to do”
- Works everywhere. Always.
Stick with click

- click is not a mouse event
- click really means “activate”
- “I am now going to use this element for the task it’s meant to do”
- Works everywhere. Always.
- But: slow. About 300ms delay between touch and the following of the link
The slowness of click

What does touching the screen mean?
- “I want to click on this element”
- “I want to scroll”
- “I want to zoom”
- “I want to hold my touch”
- “I want to double-tap”

Thus, a single touchstart event doesn’t give enough clues. The OS needs to wait a little while to figure out what you mean. Hence the delay.
Stick with click

- Fire scripts on touchstart instead of onclick.
- That works
- but now YOU are responsible for making sure the user can zoom
- and scroll
- and double-tap
- and tap-hold
- So don’t do it. Don’t try to be too clever. Accept the medium’s limitations.
It’s all about events
Point is: does every interaction mode need its own events?
So far the answer has been YES
So far the answer has been YES

Why, though?
Interaction modes

In the beginning was the mouse - and it had its own mouse events
Interaction modes

Then we discovered computers had keyboards and browsers keyboard events and we had to adjust our scripts.
Interaction modes

And then there was touch with its own events and we had to adjust our scripts again.
Separate events

<table>
<thead>
<tr>
<th>Keyboard</th>
<th>Mouse</th>
<th>Touch</th>
</tr>
</thead>
<tbody>
<tr>
<td>keydown</td>
<td>mousedown</td>
<td>touchstart</td>
</tr>
<tr>
<td>(keydown/press)</td>
<td>mousemove</td>
<td>touchmove</td>
</tr>
<tr>
<td>keyup</td>
<td>mouseup</td>
<td>touchend</td>
</tr>
</tbody>
</table>
# Integrated events

<table>
<thead>
<tr>
<th>Keyboard</th>
<th>Mouse</th>
<th>Touch</th>
</tr>
</thead>
<tbody>
<tr>
<td>keydown</td>
<td>pointerdown</td>
<td></td>
</tr>
<tr>
<td>(keydown/press)</td>
<td>pointermove</td>
<td></td>
</tr>
<tr>
<td>keyup</td>
<td>pointerup</td>
<td></td>
</tr>
</tbody>
</table>

This is Microsoft’s idea, and it merits careful consideration.

(The Microsoft events also work for pen. I haven’t tested this, though.)
Touch events

- touchstart
- touchmove
- touchend
Touch/mouse/pen events

- `pointerdown`
- `pointermove`
- `pointerup`
Example

• http://quirksmode.org/touchevents
• Drag and drop
• Works with minimal changes
el.ontouchstart = function () {
    el.ontouchmove = function () {
        ...
    }
}

el.ontouchend = function () {
    el.ontouchmove = null;
}
}
el.ontouchstart = el.onmspointerdown = function () {
    el.ontouchmove = el.onmspointermove = function () {
        ...
    }
}

el.ontouchend = el.onmspointerup = function () {
    el.ontouchmove = null;
}

}
Touch events

```
el.ontouchstart = el.onmspointerdown = function () {
  el.ontouchmove = el.onmspointermove = function () {
    ...
  }
}
el.ontouchend = el.onmspointerup = function () {
  el.ontouchmove = null;
}
```
Microsoft’s approach

It turns out that you have to add one line of CSS:

-ms-touch-action: none;
Microsoft’s approach

It turns out that you have to add one line of CSS:

-ms-touch-action: none;

And no, I don’t much like that.

(But it gets worse. Just wait.)
Microsoft’s approach

These are the -ms-touch-action values. They tell the browser which gestures are allowed:

- none - nothing allowed
- auto - everything allowed
- pan-x and pan-y - scrolling x or y
- pinch-zoom - pinch-zooming
- double-tap-zoom - double-tap zooming
- manipulation - everything except double-tap
Example

- http://quirksmode.org/touchevents
- -ms-touch-action
- Here are all the values. Play with them.
Event info

- touches array: contains all current touches
- changedTouches array: contains all touches that caused an event to fire
- targetTouches array: contains all touches on the target element
function handleTouch(event) {
    var currentTouch = event.changedTouches[0];
    var xCoor = currentTouch.clientX;
    var yCoor = currentTouch.clientY;
}
Event info

• But not in the Microsoft model

• The touches arrays don’t exist.

• Instead, we just read out the coordinates in the old-fashioned way.

• I think this is a good idea in 90% of the cases

• but in the remaining 10% we actually need a list of touch actions taking place, and there isn’t any.
function handleTouch(event) {
    var currentTouch;
    if (event.changedTouches) {
        currentTouch = event.changedTouches[0];
    } else {
        currentTouch = event;
    }
    var xCoor = currentTouch.clientX;
    var yCoor = currentTouch.clientY;
}
move

- touchmove continues firing as long as the user’s finger is on the screen
- even if it has left the element the event handler is defined on
- MSPointerMove, on the other hand, stops firing when the user’s finger leaves the element
Example

• http://quirksmode.org/touchevents

• Traditional and Microsoft move events

• Be sure to move out of the test element. The traditional events continue firing; the Microsoft ones don’t.
Gestures

- The MSPointerMove event doesn’t continue firing when you leave the element.
- But the MSGestureChange event does.
- Which brings us to gesture events ...
Gestures

• A coordinated action of two or more touches constitute a gesture. Example: pinch-zoom

• Apple (and only Apple) offers the gesturestart, gesturechange, and gestureend events that fire when a gesture takes place.

• Still, I’ve never used them, and not just because of the lack of support in other browsers. Why do we need them?
Gestures

Microsoft offers a lot of extra events:

- gesturestart, gesturechange, and gestureend
- gesturerelease
- gesturehold
- contentzoom
- and more
Gestures

I think I’m in love ...

- gesturestart, gesturechange, and gestureend
- gestureTap
- gestureHold
- contentzoom
- and more
Gestures

... except that they don’t work ...

- gesturestart, gesturechange, and gestureend
- gesturetap
- gesturehold
- contentzoom
- and more
Gestures

Well, they do, but Microsoft has gone temporarily insane here.

- gesturestart, gesturechange, and gestureend
- gesturetap
- gesturehold
- contentzoom
- and more
MS gesture events

```
el.onmsgesturestart = function () {
    ...
}
```
var MSGesture = new Gesture();
MSGesture.target = el;

el.onmsgesturestart = function () {
    ...
};
var MSGesture = new Gesture();
MSGesture.target = el;
el.onmspointerdown = function (e) {
    MSGesture.addPointer(e(pointerId);
}
el.onmsgesturestart = function () {
    ...
}
MS gesture events

```javascript
var MSGesture = new Gesture();
MSGesture.target = el;
el.onmspointerdown = function (e) {
    MSGesture.addPointer(e.pointerId);
}
el.onmsgesturestart = function () {
    ...
}
```

And don’t forget our old friend:

```
-ms-touch-action: none;
```
Gestures

... and even then contentzoom doesn’t work

• gesturestart, gesturechange, and gestureend
• gesturertap
• gesturehold
• contentzoom
• and more
Example

- http://quirksmode.org/touchevents
- Apple and MS Gestures
- Be sure to move out of the test element. MSGestureChange continues firing.
Mobile is becoming increasingly important to web designers and developers because users expect a site to work on their phones. Simultaneously, the web is becoming increasingly important to the mobile world because it is the only way to deploy an application to any phone.

Nowadays most web conferences feature a mobile session, and most mobile conferences a web session. The obvious next step is Mobilism: a conference wholly dedicated to mobile web design and development.

Just like in 2011 we’ll invite some of the best speakers from the web development and the mobile world to guide you through the confusing jumble of platforms, screen sizes, and browsers that is the mobile ecosystem.

Join our mailing list

Please leave your email address if you want to receive Mobilism 2012 information as soon as it becomes available.

Email address
Native scrolling

• I wrote the scroll script back in 2011, when there was no other way of getting the effect.

• Meanwhile there is:
  • -webkit-overflow-scrolling: touch
  • This allows one-finger native scrolling
  • But only on iOS and BB10
Native scrolling

- `-webkit-overflow-scrolling: touch`
- This allows one-finger native scrolling
- But only on iOS and BB10
Native scrolling

- `-webkit-overflow-scrolling: touch`

- This allows one-finger native scrolling

- But only on iOS and BB10

- Android, Chrome, Opera 14, BB PlayBook, IE, and Firefox do not need it. They allow native scrolling by default.
Native scrolling

- `.webkit-overflow-scrolling: touch`
- This allows one-finger native scrolling
- But only on iOS and BB10
- Android, Chrome, Opera 14, BB PlayBook, IE, and Firefox do not need it. They allow native scrolling by default.
- Then why don’t Apple and BlackBerry?
Native scrolling

• Giving an element native, inertial scrolling behaviour means part of the page needs independent movement.

• This is quite expensive in terms of memory and processor speed.

• So that’s why Apple and BlackBerry decided to only do it if the web developer explicitly asks for it.

• (Still, eventually people will give all their scrolling layers `-webkit-overflow-scroll: touch`, and the problem will reappear.)
Recap

• Stick with click

• Do we need separate events for every interaction mode?

• MS pointer events need -ms-touch-action

• Gesture events are usually not very useful, and are only supported by Safari and IE.

• -webkit-overflow-scrolling
Thank you

I’m going to put these slides online.

Questions?

Peter-Paul Koch
http://quirksmode.org
http://twitter.com/ppk

Mobilism, 16 and 17 May 2013