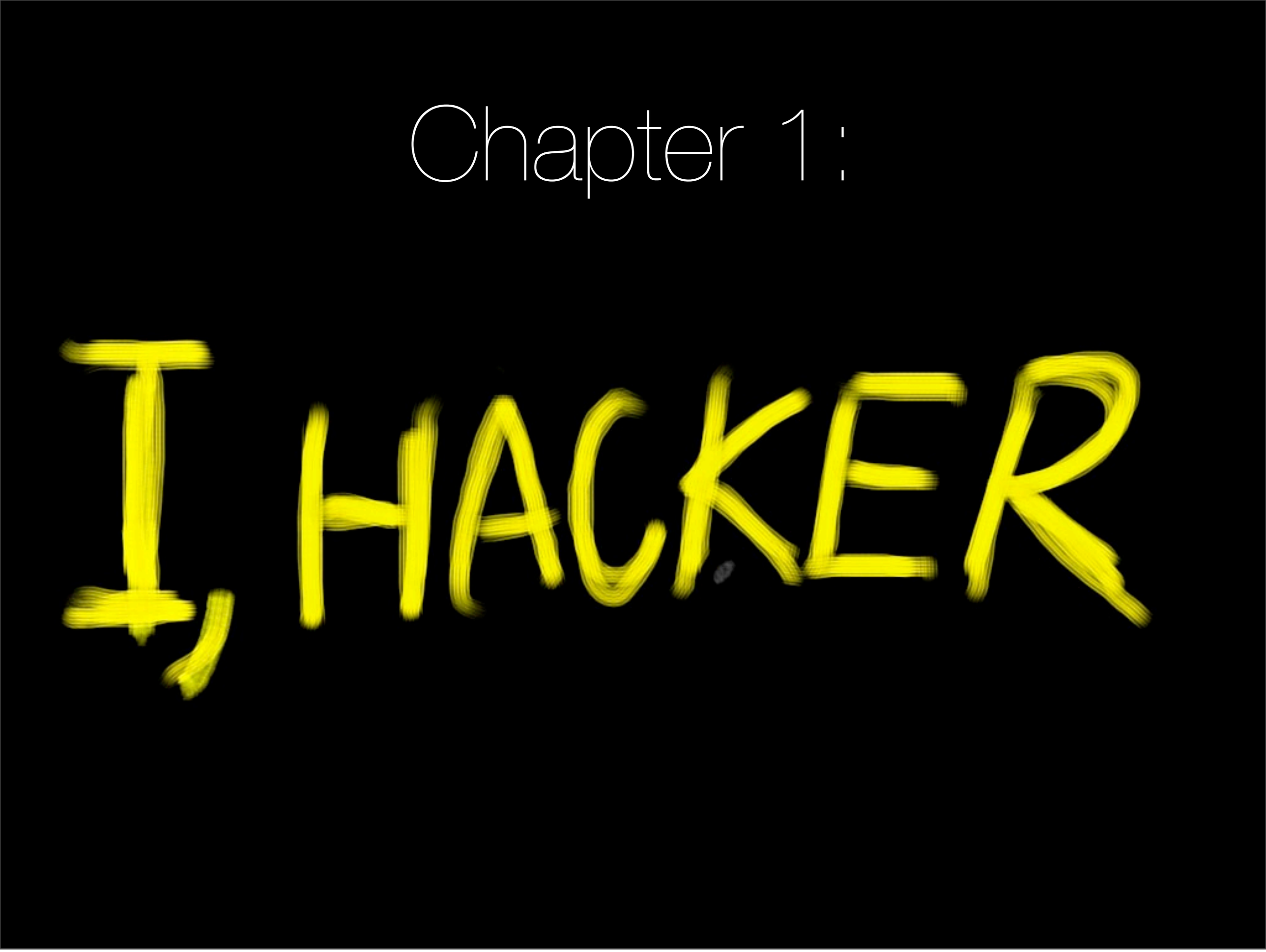
FINGER TIPS

Lessons Learned From Building a Touch-Based Experience

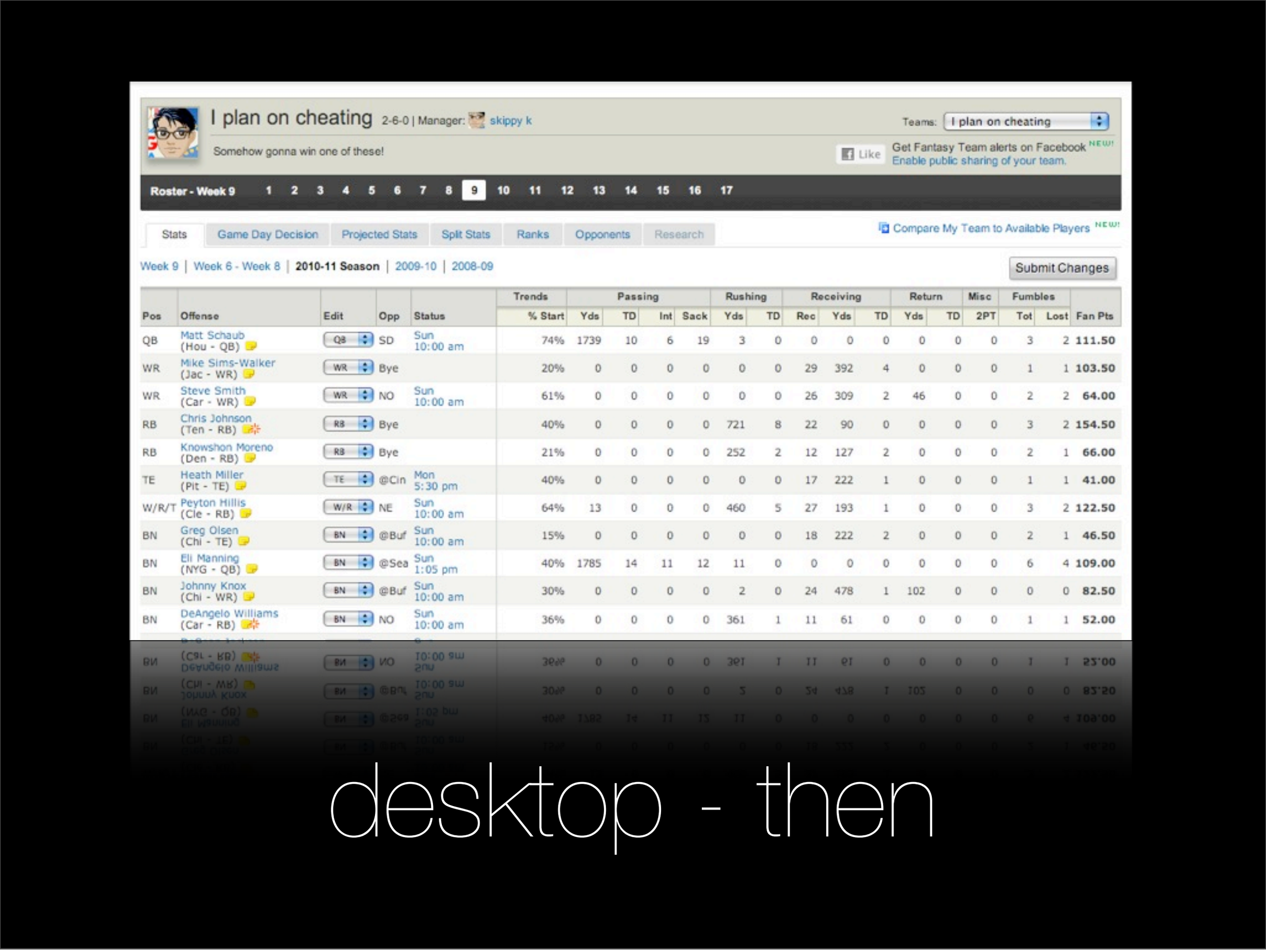
Presented November 9, 2010 at YUIConf in Sunnyvale, CA.

note: filenames in these notes refer to demonstration files used during presentation. They can be found at: https://github.com/skippykawakami/fingertips

Chapter 1 :

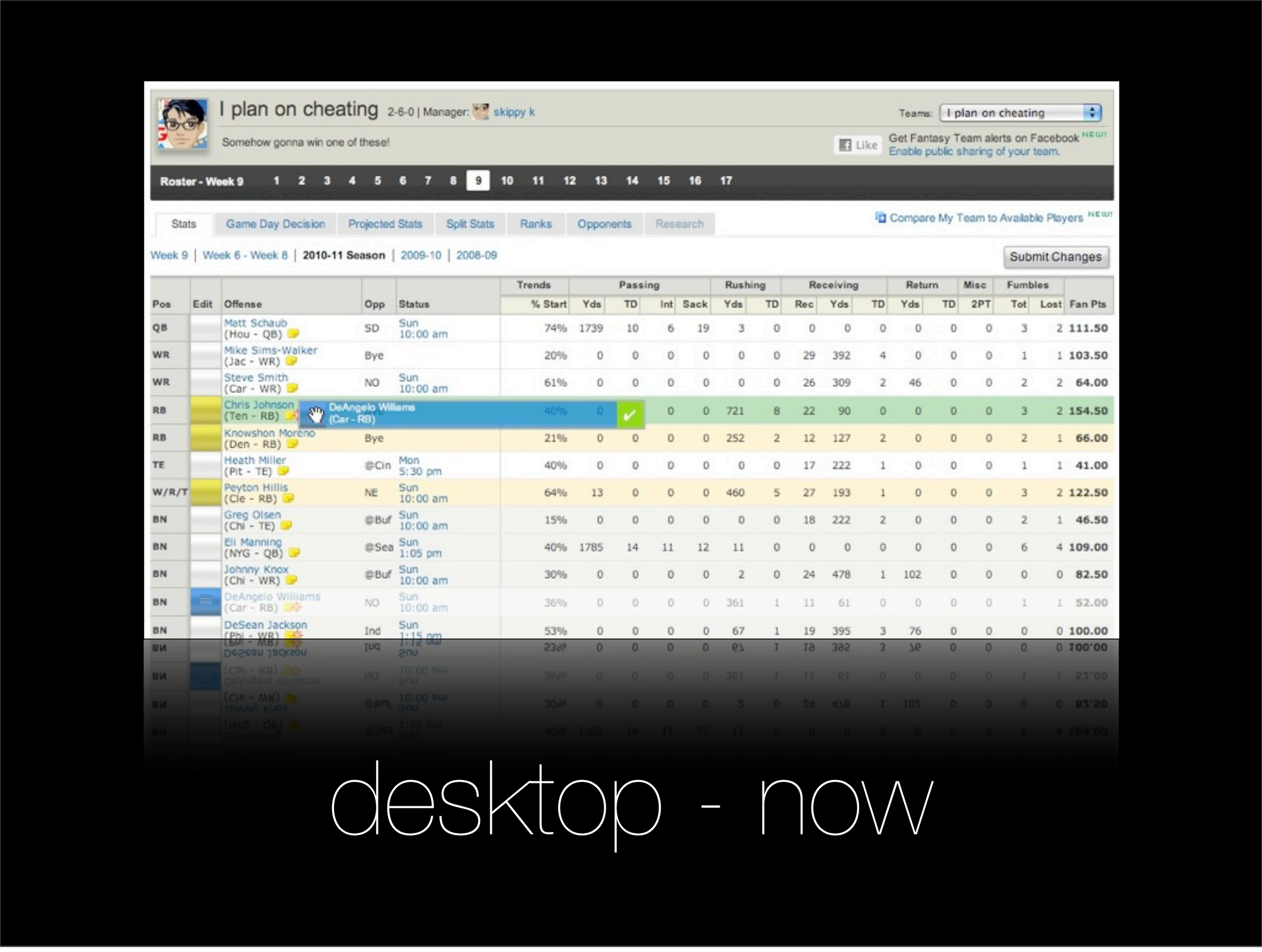


YUI 3.2 makes it easy to add touch interactions to your web apps. But as it turns out, making a great touch experience means more than just updating from YUI 3.1 to YUI 3.2.

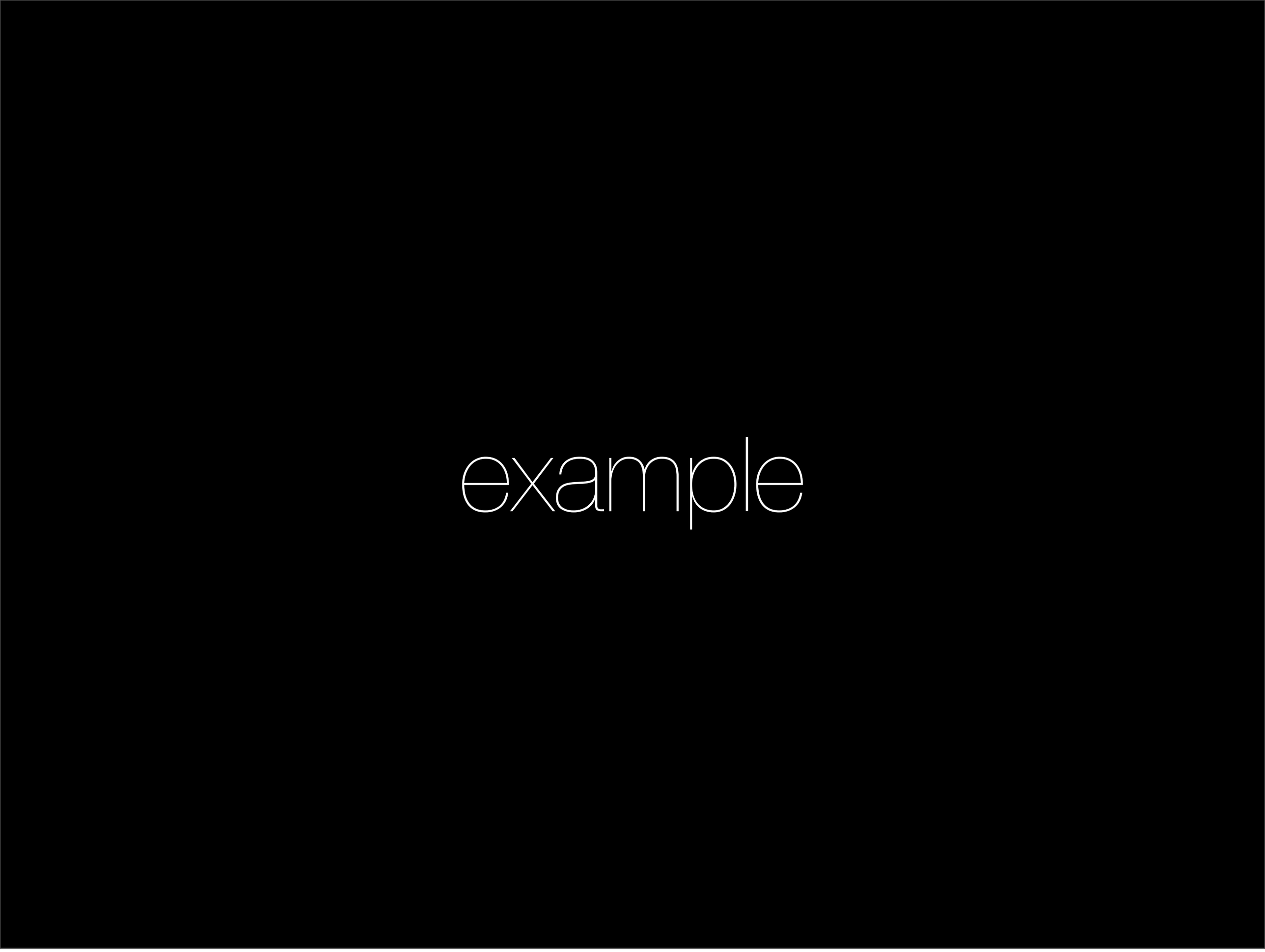


desktop - then

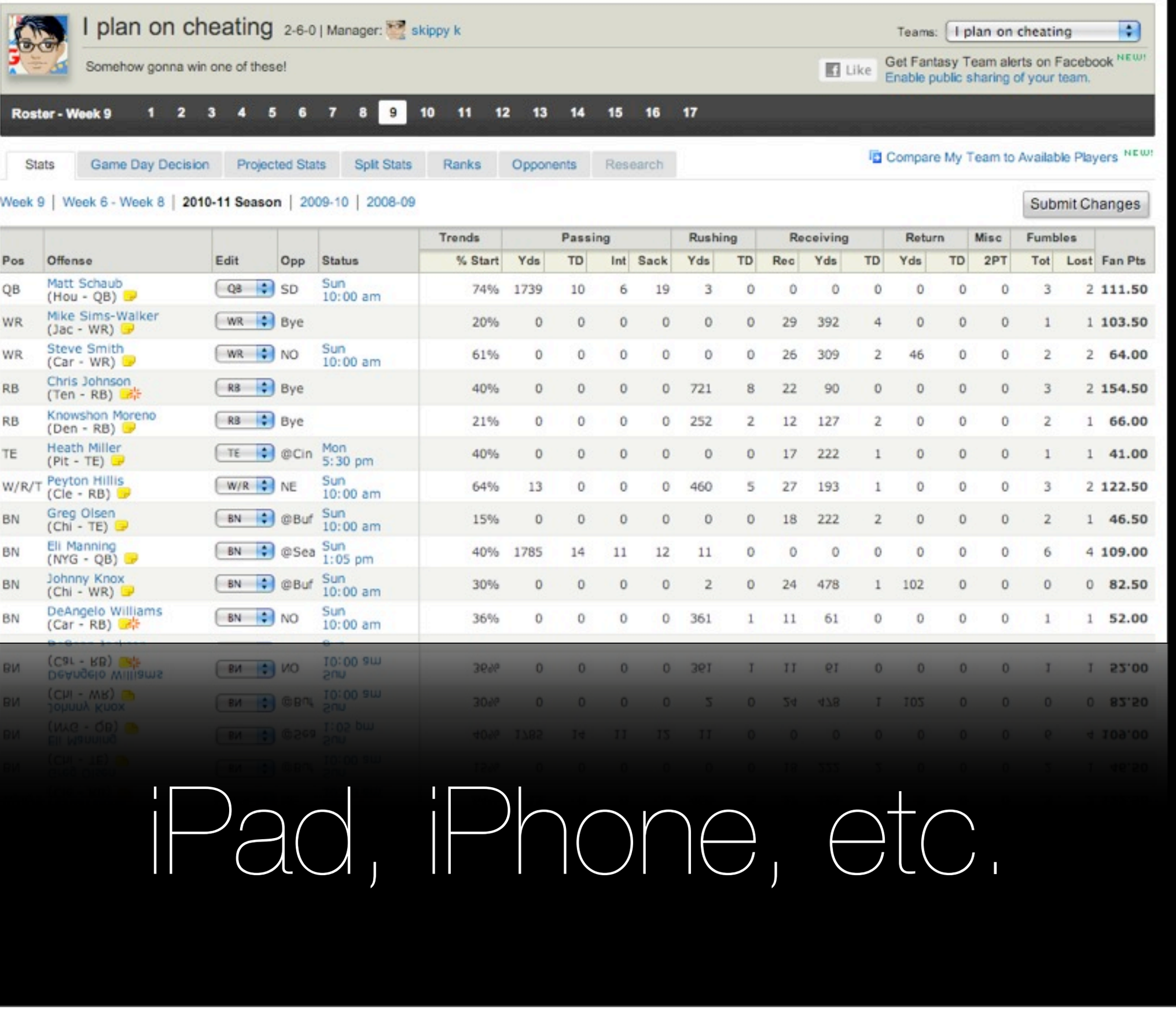
Old style rosters, before drag and drop.



Drag and Drop interaction first created as a hack day project way back in the day. They're awesome.



Show Drag and Drop rosters on the desktop.



iPad, iPhone, etc.

The iOS version didn't support our Drag and Drop interactions, so user's only option was the "Classic" mode. Not cool, so for our three-day Hacktackular event this year, one of our projects was to make the rosters touch compatible. This involved rewriting the DnD Roster code from YUI 2 to YUI 3, and creating a touch library for YUI 3.1 (this work predated the release of YUI 3.2).

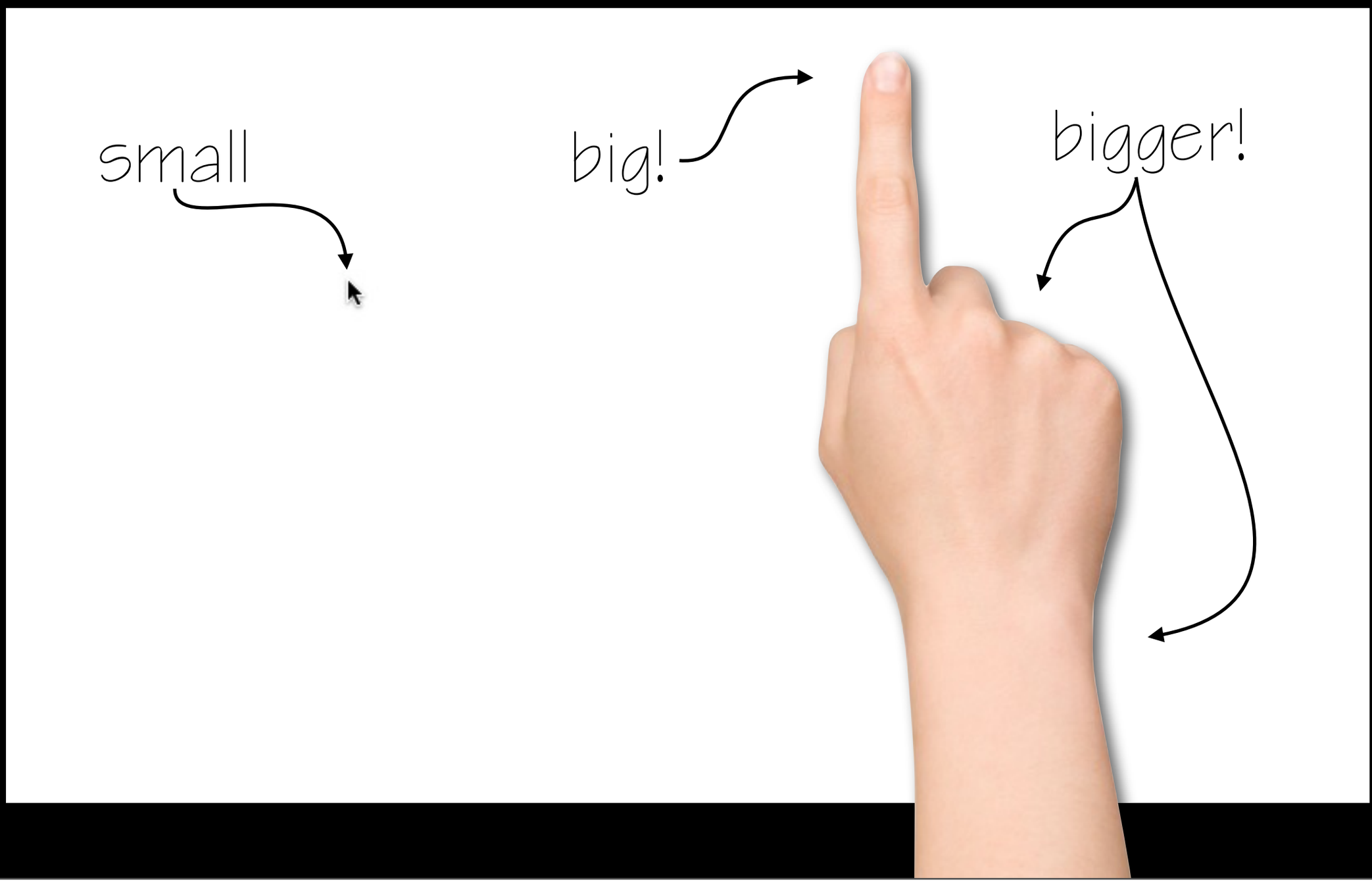
GOAL: Recreate the desktop experience for touch devices

This was our goal heading into the project. But as it turns out, there are problems with this goal as I found out almost immediately. It’s a story I like to call...

Chapter 2:

MY BIG, DUMB FINGER

Cursors vs. Fingers



You have to consider not only the size of the fingertip touching the device, but the rest of the hand and the arm, and what that might be obscuring.

example

1. Default top-left means your hand hides the item you’re dragging (bigdumbfinger.html)
2. What we have on the desktop doesn’t work. (bigdumbfinger-2.html)
3. Centering or auto-offsetting don’t work: what your dragging is more visible, but there are other problems. (bigdumbfinger-3.html)
4. So we do something different for touch devices. (bigdumbfinger-4.html)

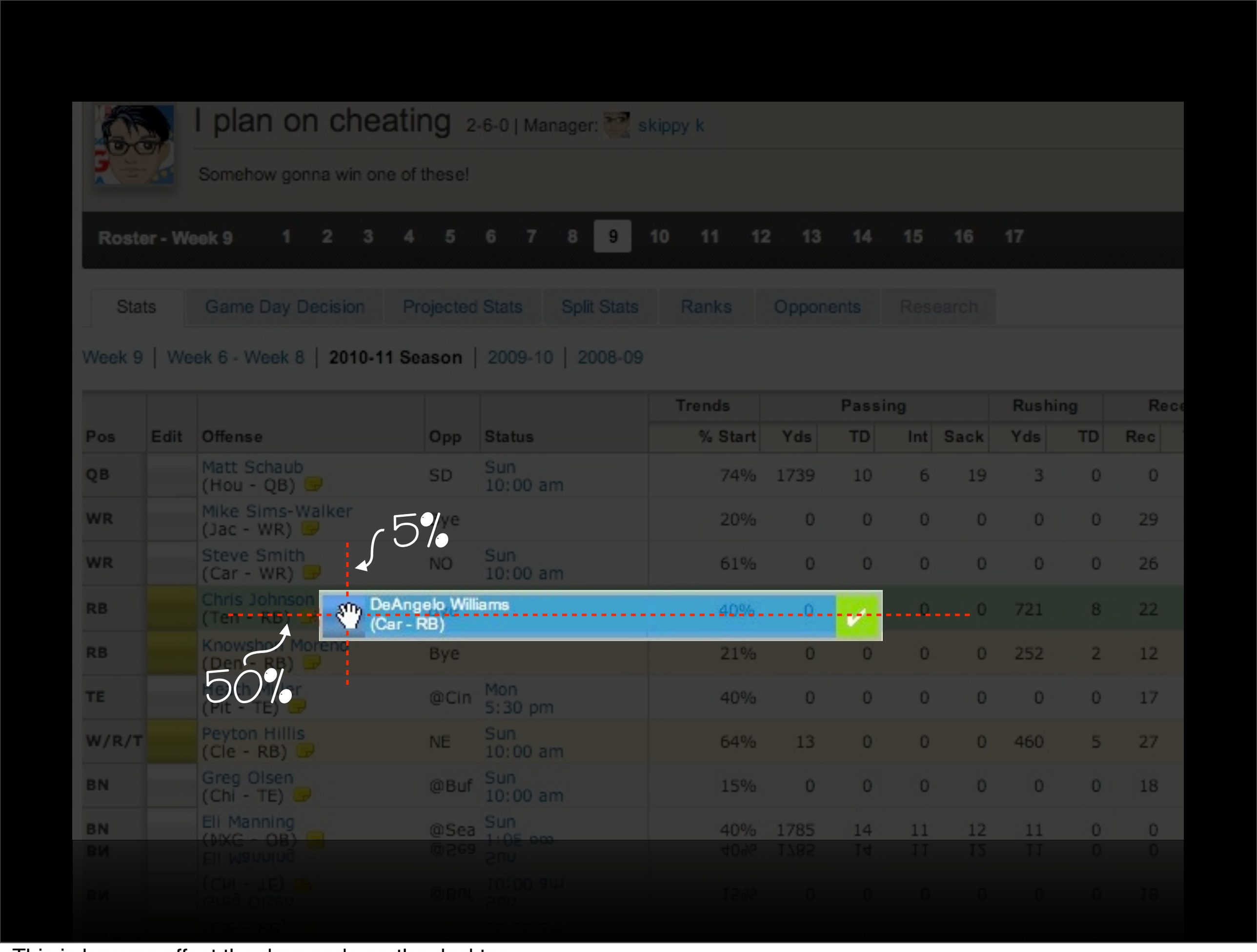
Drag Offsetting

```
    var drag = new Y.DD.Drag({
      node: n,
      startCentered: true,
      offsetNode: true,
      groups: []
}).plug( Y.Plugin.DDDragOffset, {
      offsetXY: [0.05, 0.5],
      touchOffsetXY: [0.35, 0.5]
});
```

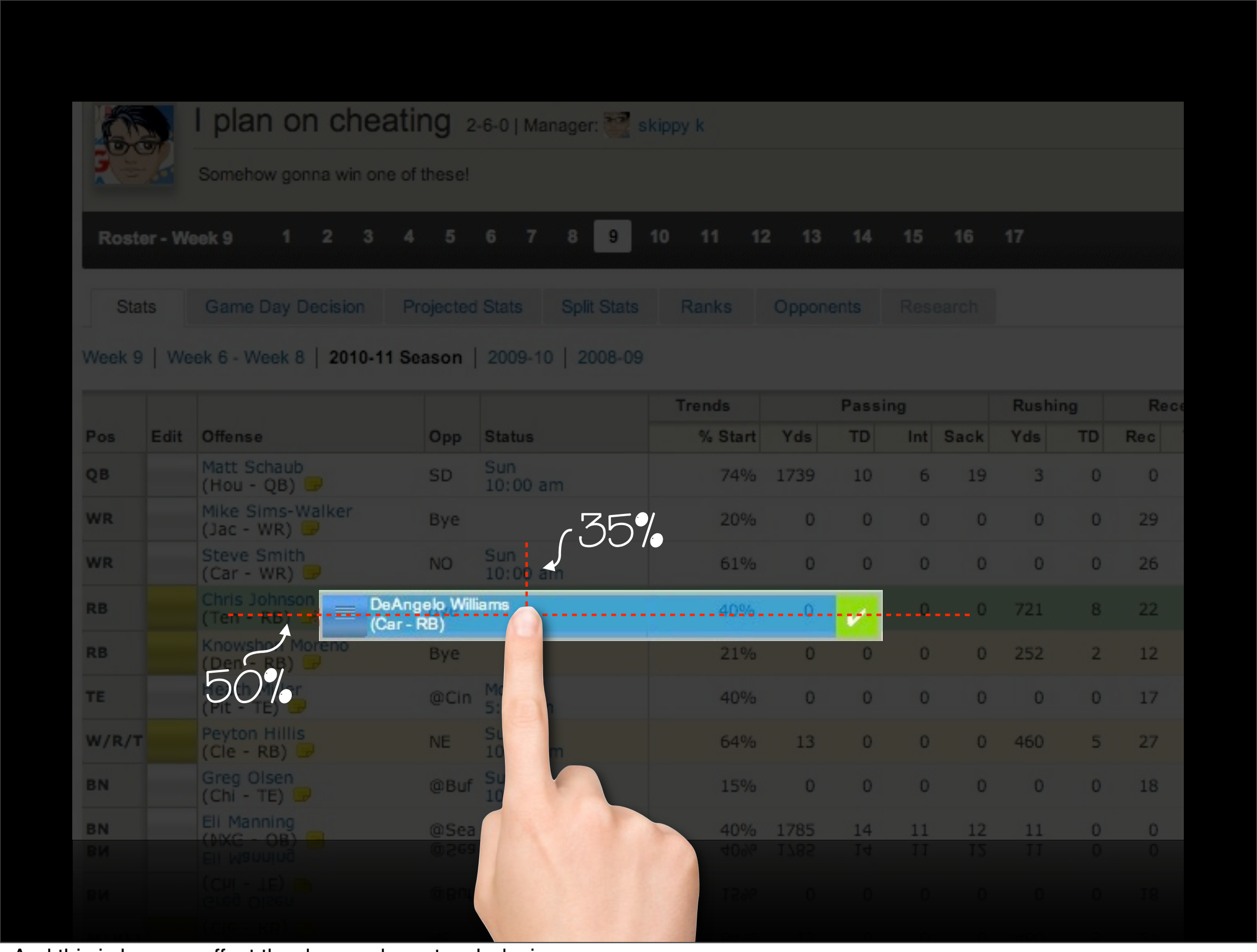
For mice

For touch devices

Y.Plugin.DDDragOffset is a plugin we developed to allow custom offsetting in drag operations, and to allow for a different amount of offsetting for touch devices. Source code is in github repository.



This is how we offset the drag node on the desktop...



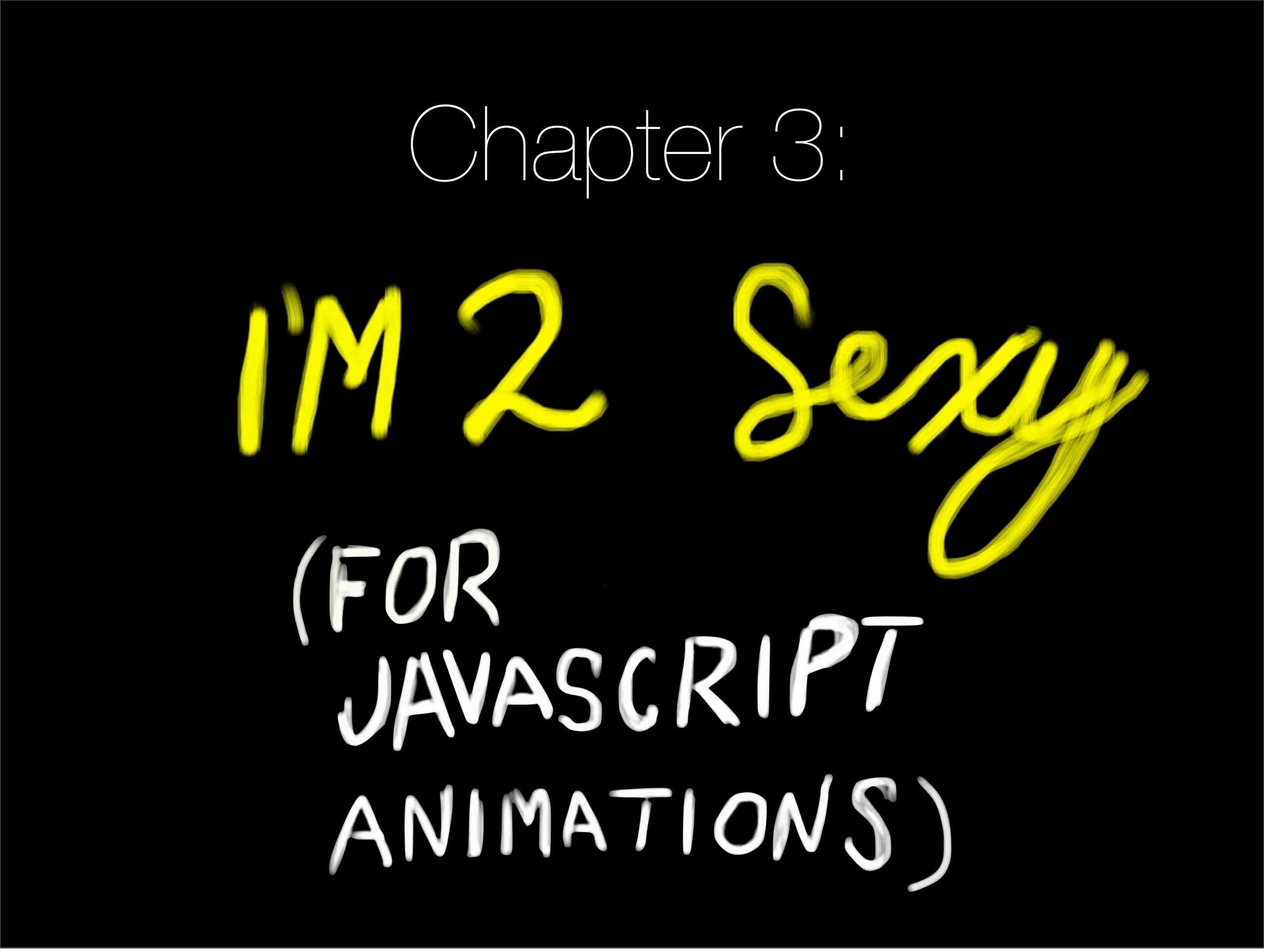
And this is how we offset the drag node on touch devices.

finger tip 1:

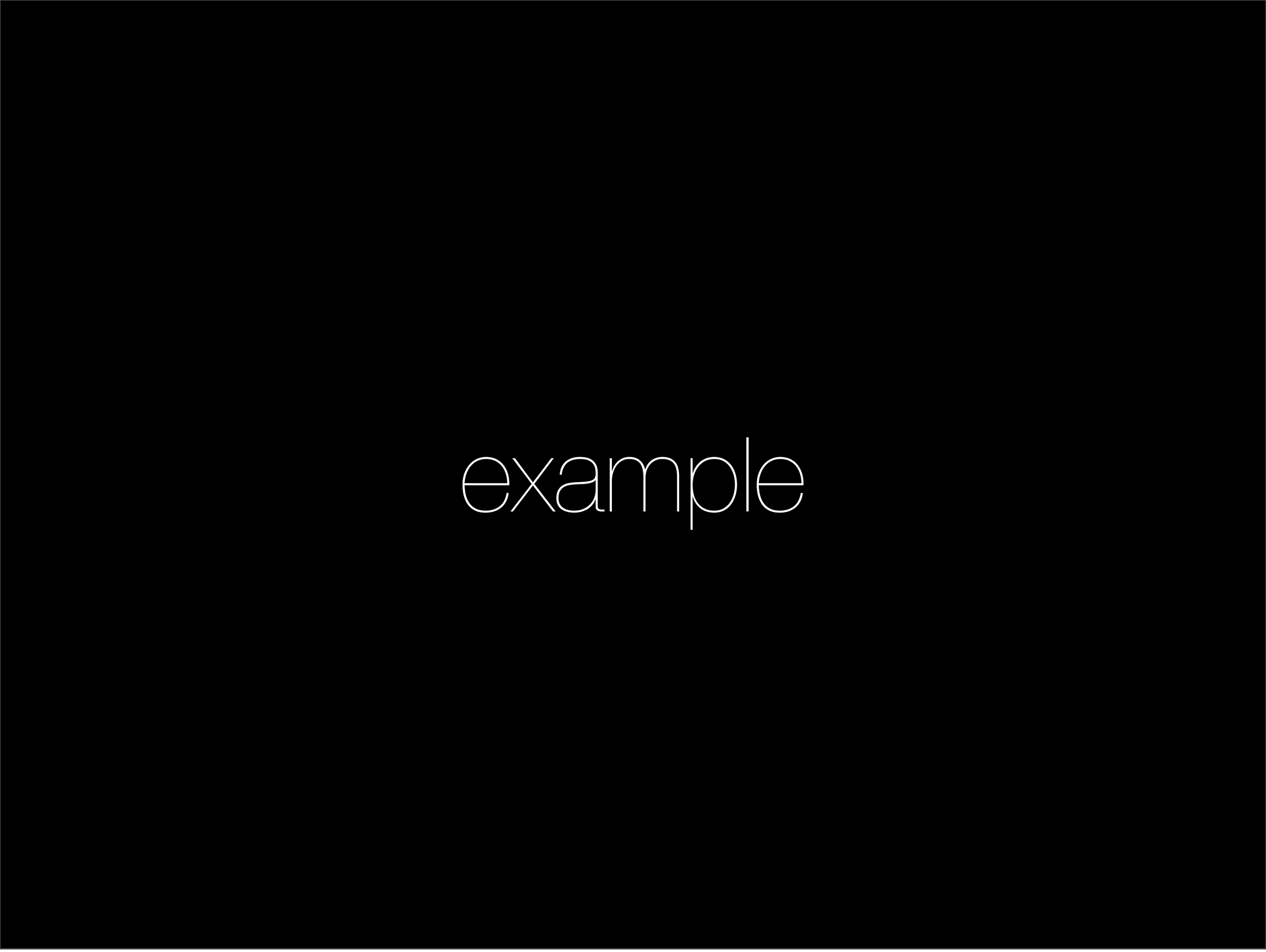
The Touch experience is **similar** but **not identical** to the Desktop

This is probably the most important lesson, which is why it’s first. The size, the speed and the interaction model of the iPad or the iPhone is different than the desktop. Sometimes you have to recognize that what works on the desktop will not work on a different device.

Chapter 3:



Actually, I'm (obviously) not 2 sexy for anything. But the iPad is.



1. Javascript animations are slow in iOS devices. (animation.html)
2. Most CSS-based animations are also slow! (animation-2.html)
3. Except when using -webkit-transform and opacity only, because they’re hardware accelerated. (animation-3.html)
4. JS and CSS animations are even worse when you have properties like box-shadow, but hardware acceleration is still smooth. (animation-4.html)
5. Simultaneous animations choke almost completely on iOS devices, unless they’re hardware accelerated. (animation-5.html)
6. Various examples of what can be done with -webkit-transform.

CSS Animations

```
.css-accelerated.animate {
    -webkit-animation-name: accelanim;
    -webkit-animation-duration: 1s;
    -webkit-animation-iteration-count: 1;
    -webkit-transform-origin: 0 0;
    opacity: 0;
}

@-webkit-keyframes accelanim  {
    from {
        -webkit-transform: scale(1);
        opacity: 1;
    }

    to {
        -webkit-transform: scale(4);
        opacity: 0;
    }
}
```

Example of syntax for CSS animations (as opposed to CSS transitions). CSS animations are nice because they generate events at the beginning, end and in between iterations of animations, as well as offering more control over when things animate.

Slow CSS Animations

```
@-webkit-keyframes cssanim  {
    from {
        width: 200px;
        height: 200px;
        opacity: 1;
    }

    to {
        width: 800px;
        height: 800px;
        opacity: 0;
    }
}
```

these cause slow reflow calculations

Most CSS properties are too expensive for JS on iOS to recalculate efficiently.

Fast CSS Animations

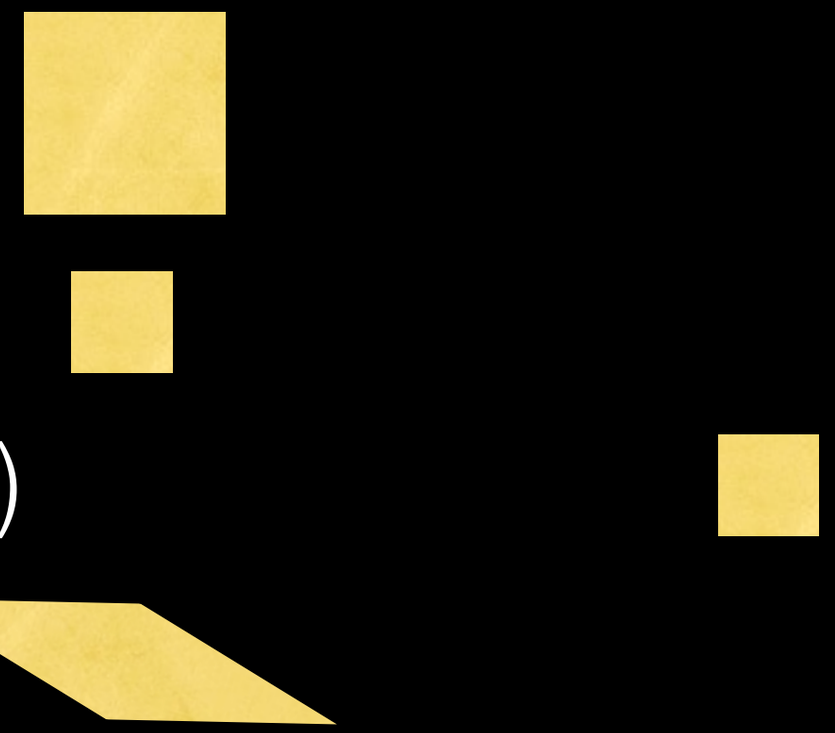
```
@-webkit-keyframes cssanim  {
    from {
        -webkit-transform: scale(1);
        opacity: 1;
    }

    to {
        -webkit-transform: scale(0);
        opacity: 0;
    }
}
```

don't reflow, and can be handled as an image

-webkit-transform and opacity are handled by hardware acceleration by making an image (actually an OpenGL polygon, I've been told) and transforming that. Because they don't affect page layout, they won't trigger reflows.

Accelerated CSS Properties

- opacity
- -webkit-transform:
  - scale()
  - rotate()
  - translate()
  - skew()



Brief intro to what the various transformations do. There are also 3D equivalents of many properties. You really have to see the slides in action to see the effect.

finger tip 2:

sexy can slow you down.

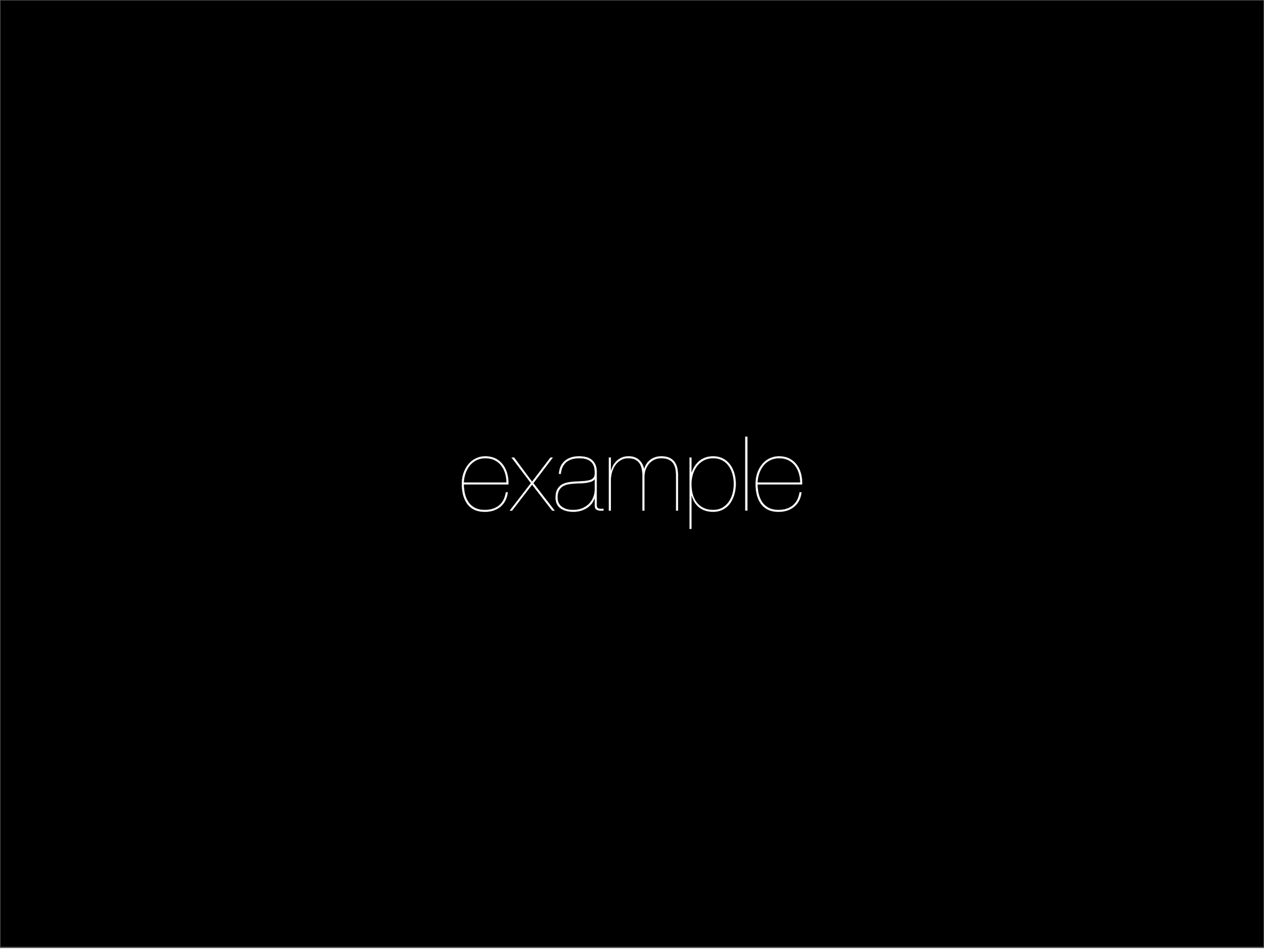
sexier can speed you up.

But don’t go nuts. A little goes a long way.

Chapter 4:

I LEARN TO PROCRAST

Sorry, didn't get around to finishing this slide.
Procrastination: Bad for your performance review. Good for your performance.



Waiting for code to finish initializing delays the page from being ready for the user. (procrastination.html)

Don't dive into the DOM until you have to

```
/* SLOWER: */

var draggables = Y.all("div.draggable");

// ... snip ...

Y.one("#animation-thing").addClass("animate");

draggables.each(function(n) {
    dragCreator.makeRowDraggable(n);
});
```

Doing CSS queries too soon can unnecessarily delay style and DOM changes.

Don't dive into the DOM until you have to

```
/* FASTER: */

// ... snip ...

Y.one("#animation-thing").addClass("animate");

var draggables = Y.all("div.draggable");
draggables.each(function(n) {
    dragCreator.makeRowDraggable(n);
});
```

Delay CSS queries until after DOM changes if you want to see those DOM changes reflected on the page sooner -- be careful, this can potentially make things worse. In this case, it makes things a little better.

example

Do stuff Y.later()

```
/* SLOWER: */

// ... snip ...

Y.one("#animation-thing").addClass("animate");

var draggables = Y.all("div.draggable");
draggables.each(function(n) {
    dragCreator.makeRowDraggable(n);
});
```

Here we have to wait until all the drag objects have been created before the browser can return control to the user. Creating drag objects is expensive, so this effect is very noticeable.

Do stuff Y.later()

```
/* FASTER: */

// ... snip ...

Y.one("#animation-thing").addClass("animate");

var draggables = Y.all("div.draggable");
draggables.each(function(n) {
    Y.later(50, dragCreator, dragCreator.makeRowDraggable, n);
});
```

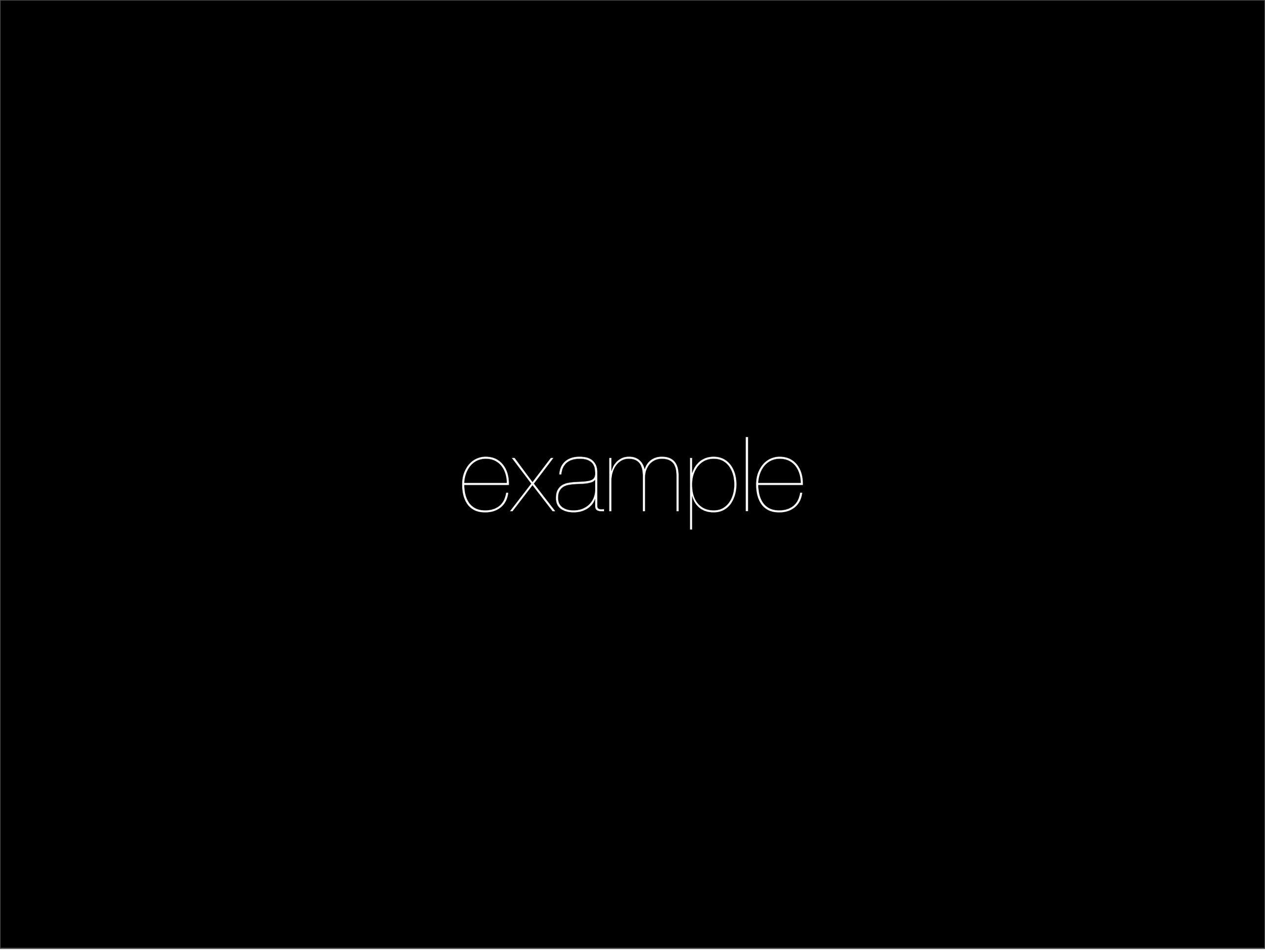
Using Y.later to delay drag creation (and divide it up into an interruptible sequence of tasks) dramatically improves responsiveness.

example

...or don't do it at all

```
/* SLOWER: */

var drag = new Y.DD.Drag({ ... cfg ... });
// ... repeat 20 times ...



/* FASTER: */

var delegate = new Y.DD.Delegate({ ... cfg ... });
// ... just once!
```

Drag objects can be delegated. This is even better, it creates nearly instantaneous initialization.


example

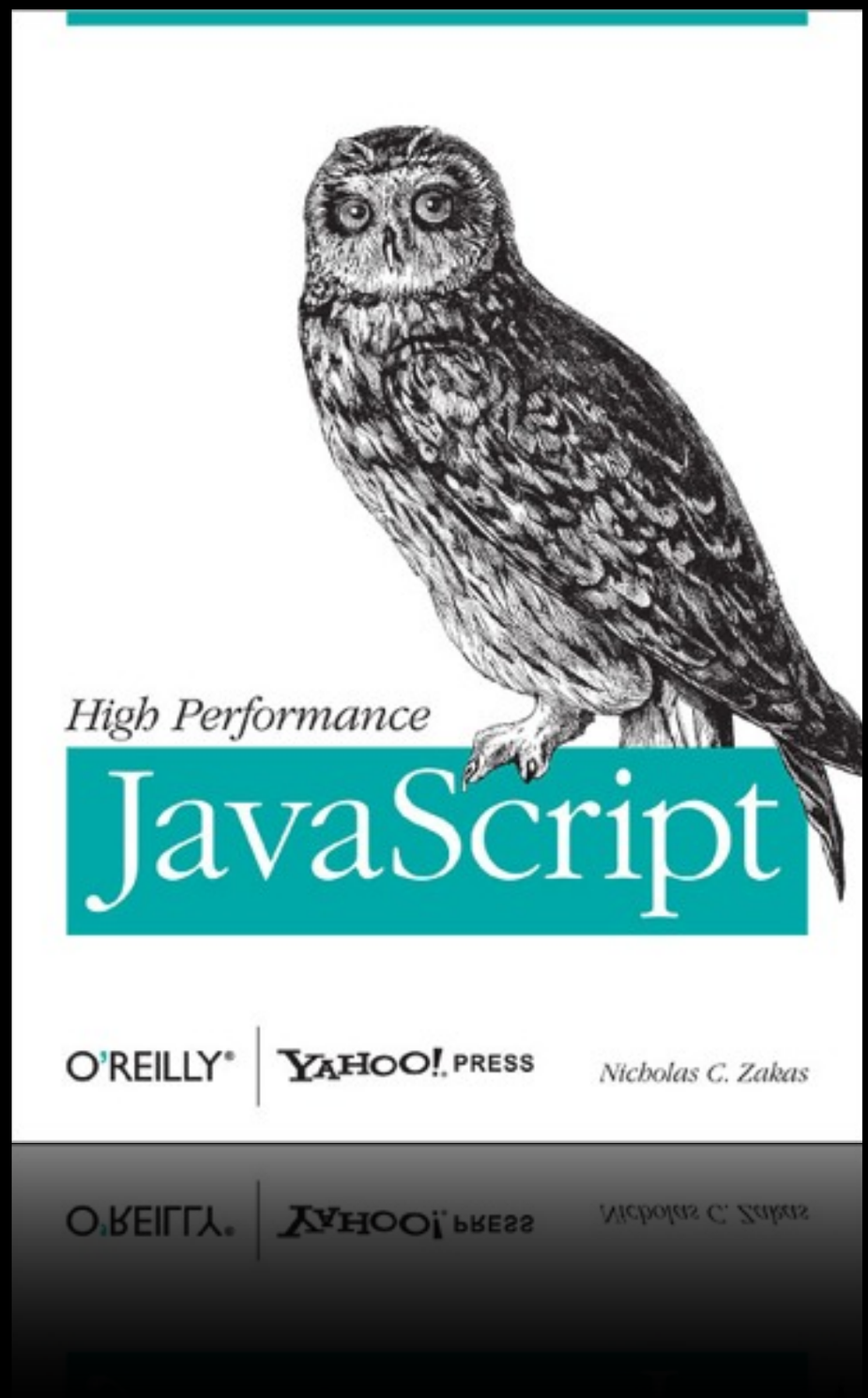
finger tip 3:

When you can't make it **faster**, make it more **responsive.**

Users don't care how fast your site is, they care how responsive it is. Improving speed is the best route to improving responsiveness, but it's not the only one.

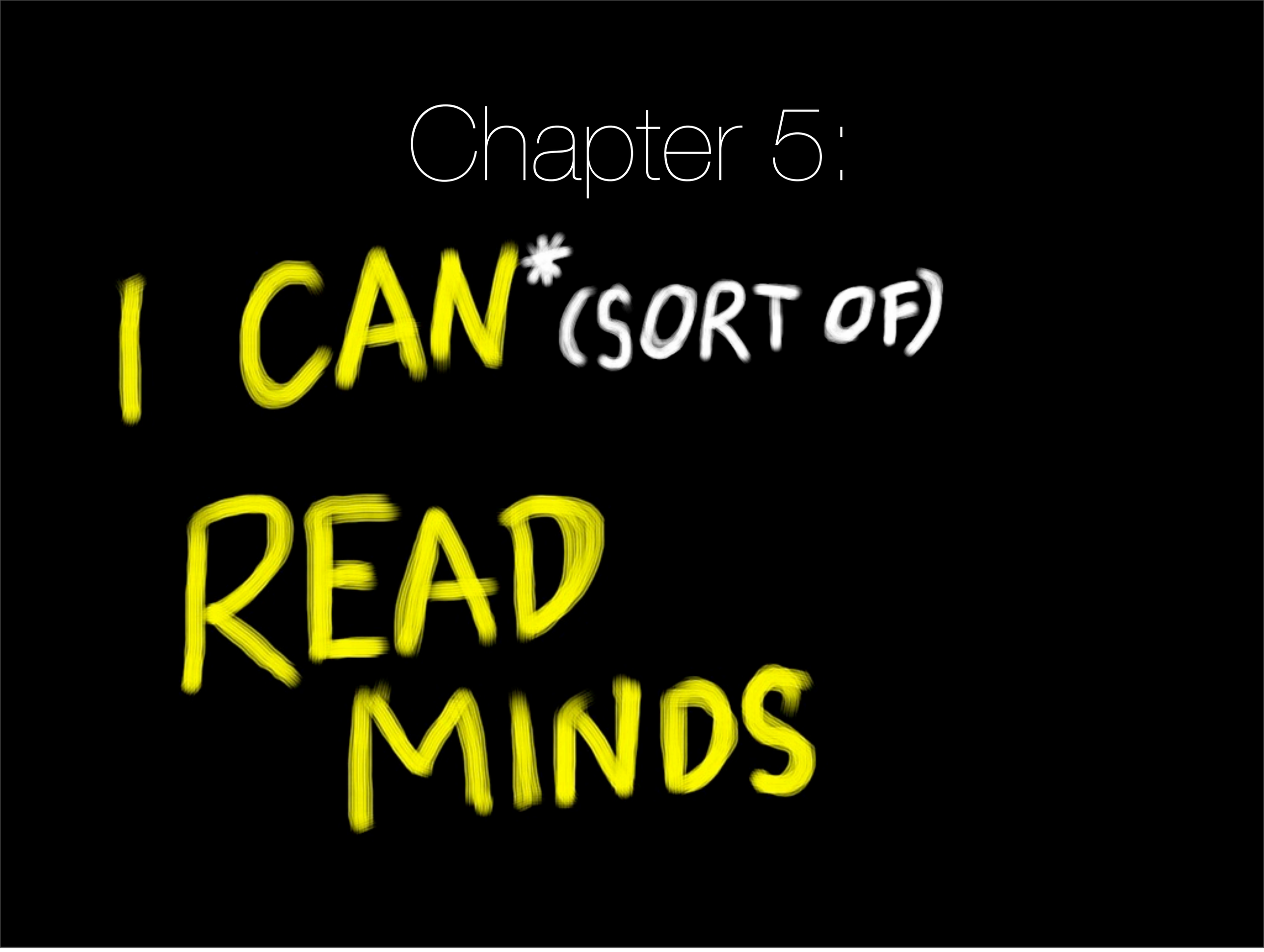
bonus finger tip:

Read this book

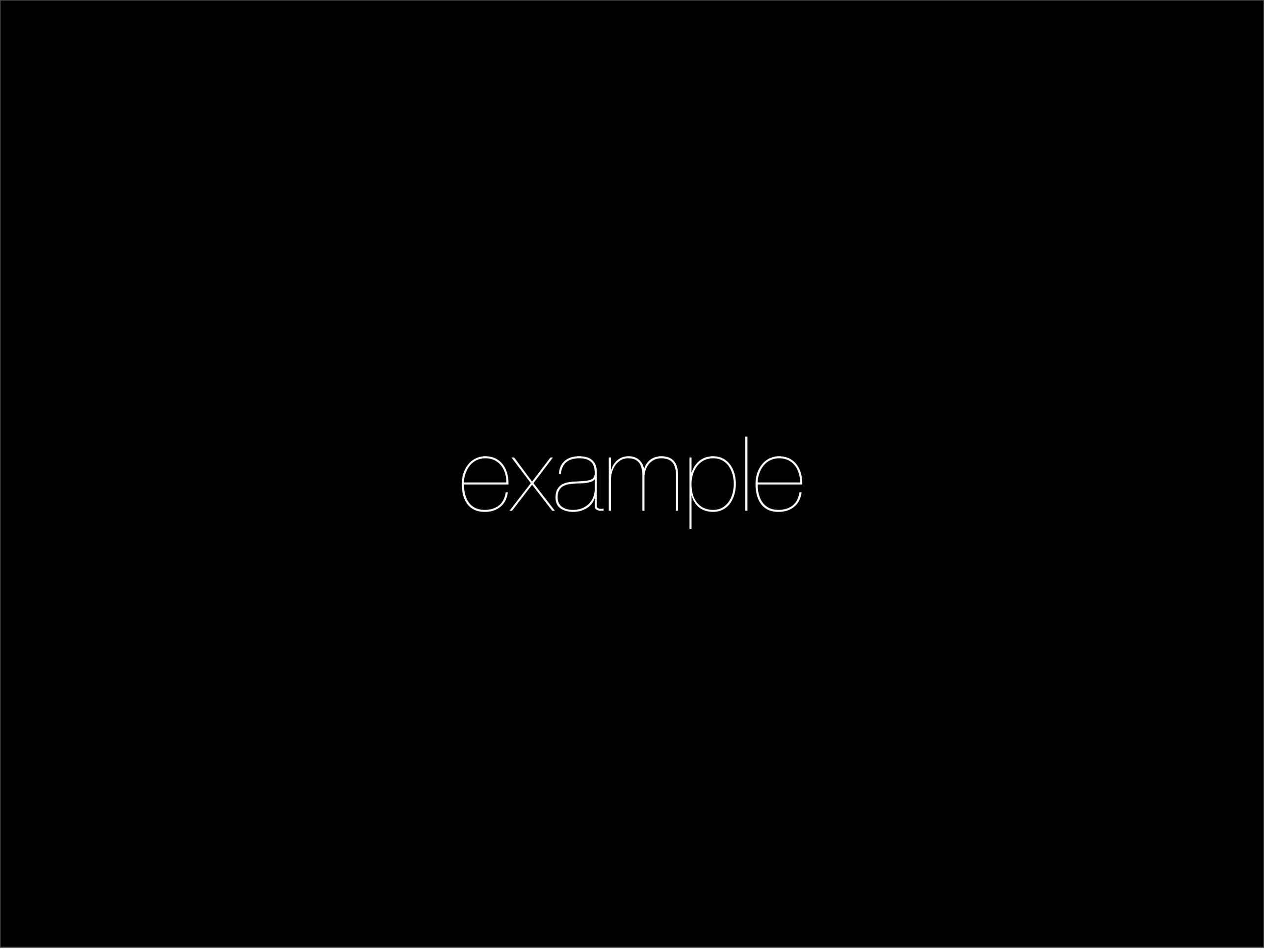


...twice.

Chapter 5:

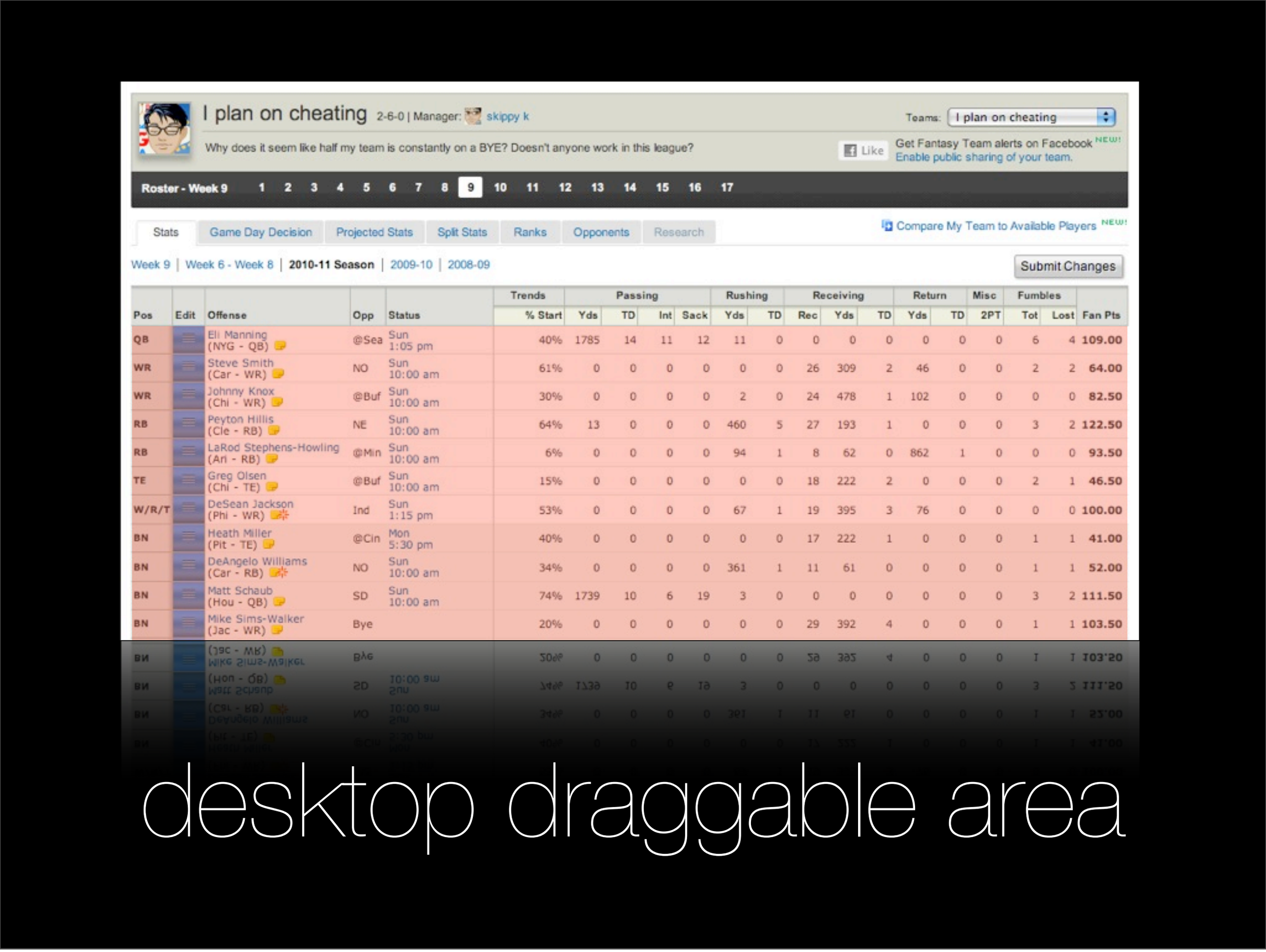


[citation needed]



The drag and scroll gestures are very similar: user places finger on touch screen and moves it around. As a result, you can’t scroll when you’re touching a draggable object. In our case, this was a serious problem. (scrolling.html)

Frustration is the gap between what a user **expects** to happen and **what actually** happens.



desktop draggable area

On the desktop, users can start a drag from anywhere in a roster row.

I plan on cheating 2-6-0 | Manager: skippy k

Teams: I plan on cheating

Why does it seem like half my team is constantly on a BYE? Doesn't anyone work in this league?

Like Get Fantasy Team alerts on Facebook NEW!
Enable public sharing of your team.

Roster - Week 9 1 2 3 4 5 6 7 8 **9** 10 11 12 13 14 15 16 17

Stats | Game Day Decision | Projected Stats | Split Stats | Ranks | Opponents | Research

Compare My Team to Available Players NEW!

Week 9 | Week 6 - Week 8 | **2010-11 Season** | 2009-10 | 2008-09

Submit Changes

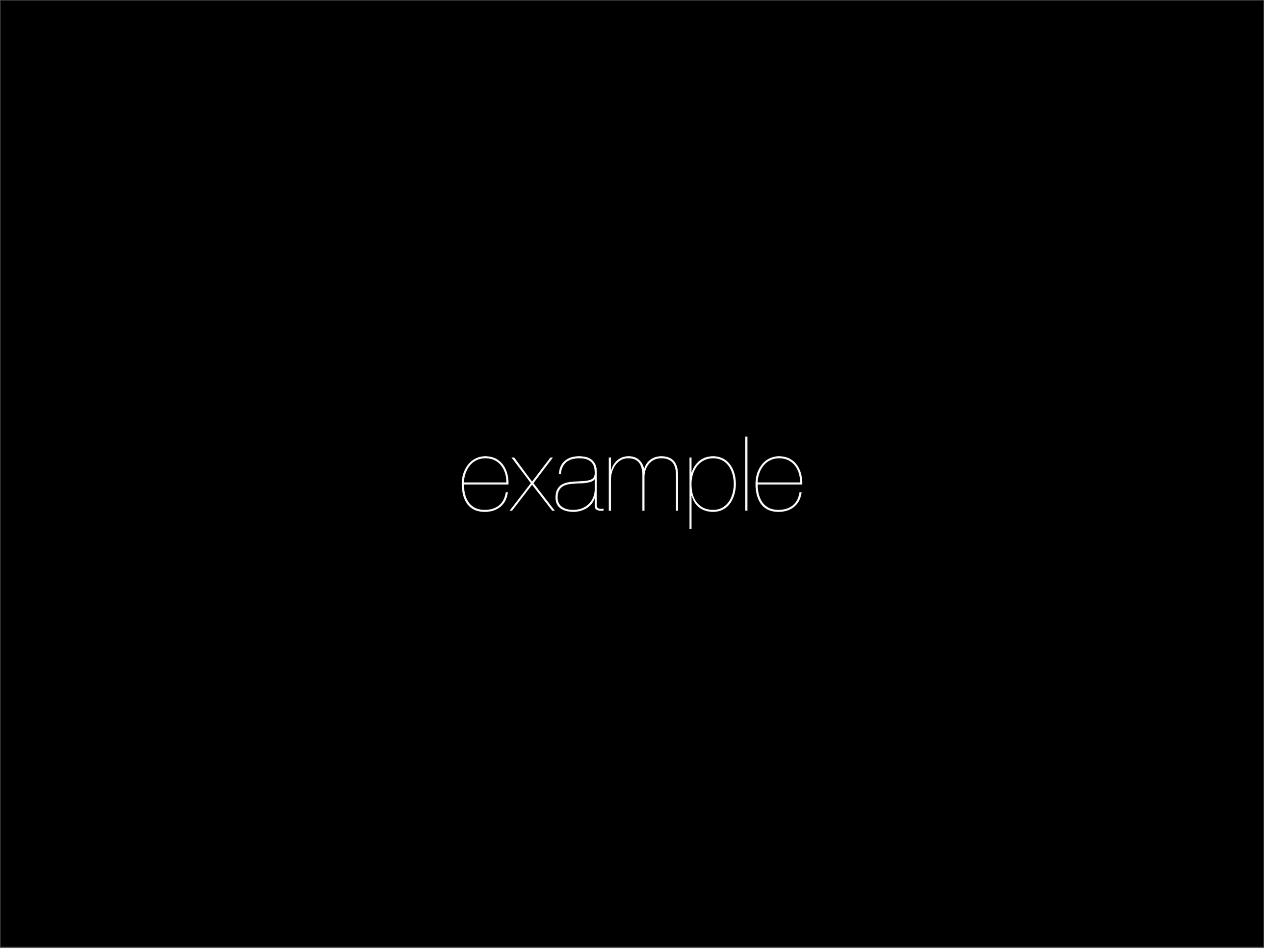
					Trends	Passing				Rushing		Receiving			Return		Misc	Fumbles		
Pos	Edit	Offense	Opp	Status	% Start	Yds	TD	Int	Sack	Yds	TD	Rec	Yds	TD	Yds	TD	2PT	Tot	Lost	Fan Pts
QB		Eli Manning (NYG - QB)	@Sea	Sun 1:05 pm	40%	1785	14	11	12	11	0	0	0	0	0	0	0	6	4	**109.00**
WR		Steve Smith (Car - WR)	NO	Sun 10:00 am	61%	0	0	0	0	0	0	26	309	2	46	0	0	2	2	**64.00**
WR		Johnny Knox (Chi - WR)	@Buf	Sun 10:00 am	30%	0	0	0	0	2	0	24	478	1	102	0	0	0	0	**82.50**
RB		Peyton Hillis (Cle - RB)	NE	Sun 10:00 am	64%	13	0	0	0	460	5	27	193	1	0	0	0	3	2	**122.50**
RB		LaRod Stephens-Howling (Ari - RB)	@Min	Sun 10:00 am	6%	0	0	0	0	94	1	8	62	0	862	1	0	0	0	**93.50**
TE		Greg Olsen (Chi - TE)	@Buf	Sun 10:00 am	15%	0	0	0	0	0	0	18	222	2	0	0	0	2	1	**46.50**
W/R/T		DeSean Jackson (Phi - WR)	Ind	Sun 1:15 pm	53%	0	0	0	0	67	1	19	395	3	76	0	0	0	0	**100.00**
BN		Heath Miller (Pit - TE)	@Cin	Mon 5:30 pm	40%	0	0	0	0	0	0	17	222	1	0	0	0	1	1	**41.00**
BN		DeAngelo Williams (Car - RB)	NO	Sun 10:00 am	34%	0	0	0	0	361	1	11	61	0	0	0	0	1	1	**52.00**
BN		Matt Schaub (Hou - QB)	SD	Sun 10:00 am	74%	1739	10	6	19	3	0	0	0	0	0	0	0	3	2	**111.50**
BN		Mike Sims-Walker (Jac - WR)	Bye		20%	0	0	0	0	0	0	29	392	4	0	0	0	1	1	**103.50**

touch draggable area

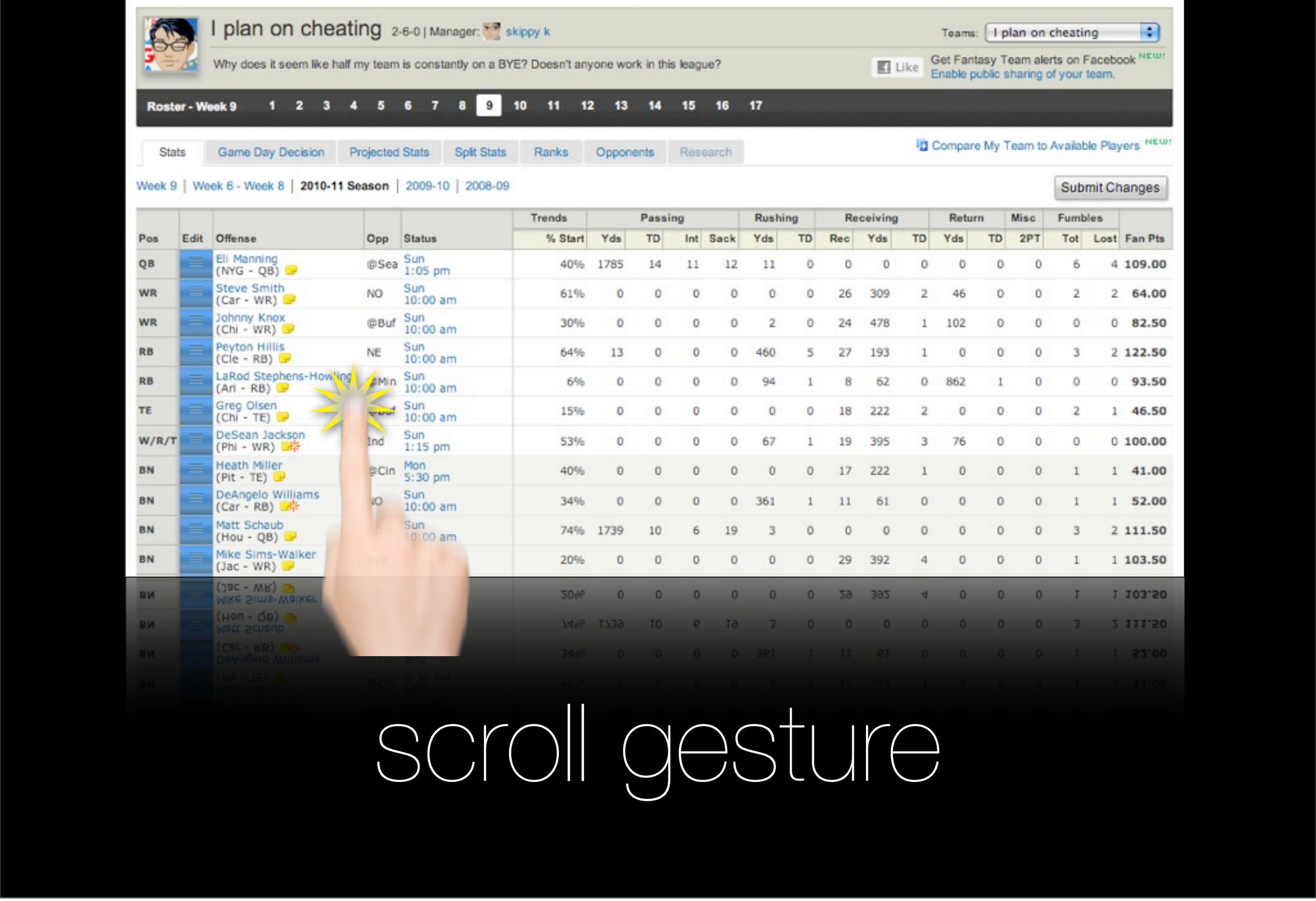
First we tried reducing the draggable area for touch devices. This helped, but didn't solve the main issue, especially when the user was zoomed into the page. (scrolling-2.html)

Frustration is the gap between what a user **expects** to happen and **what actually** happens.

Worth repeating.

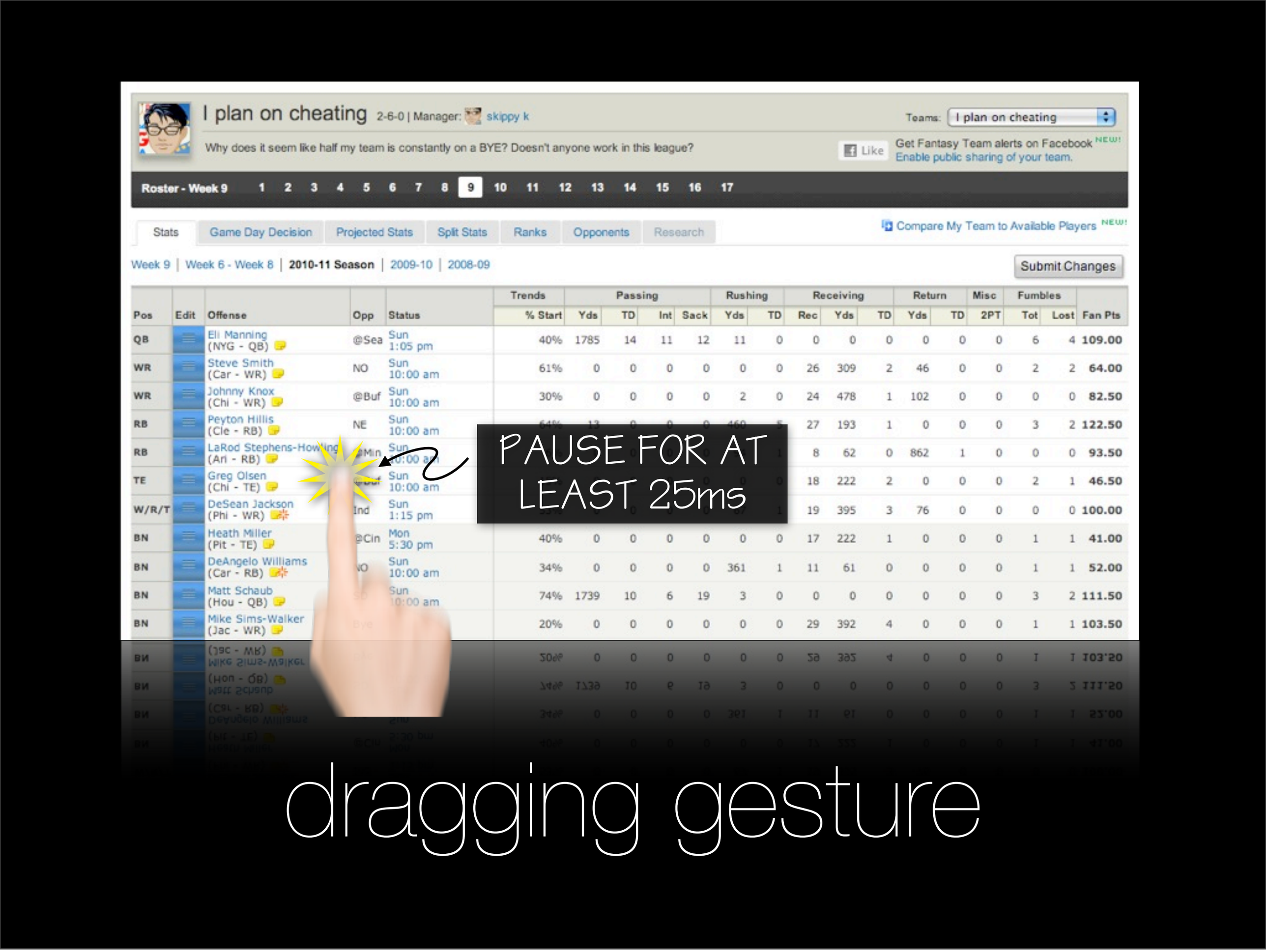


Better solution: try to determine whether the users intent is to drag or to scroll. We do this by a tiny bit of timing. (scrolling-4.html)



scroll gesture

Scroll if the user's finger doesn't pause while moving ("doesn't pause" means a touchmove event within 25ms of touchstart. You have to allow for a tiny bit of pausing. 25ms isn't set in stone, but it seemed to work the best in informal testing).



dragging gesture

We only allow the drag to start if you pause for at least 25ms. So we're using tiny, subtle variances in user action to determine user intent, which is pretty close to mind reading. Actually, technically it's a variation on "cold-reading".

But it's imperfect at best...

finger tip 4:

Reduce frustration with **E.S.P.**

Eliminate frustration with **great UI**.

... much better is to have a UI that doesn’t require psychic powers, by clearly indicating to the user what areas they can expect drag interactions to occur, and what areas they can expect scroll interactions to occur.

Resources

(just do a web search for them)

- Safari Visual Effects Guide
- Safari Web Content Guide
  - (esp. section “Handling Events”)
- Safari DOM Additions Reference

These resources provide documentation for many of the techniques we used. The YUI documentation is, of course, also invaluable.

finger tip 1:

The Touch experience is **similar** but **not identical** to the Desktop

Let’s take a closer look at this tip...

What can you do if you can skip the Desktop?

I’m not advocating skipping the desktop as a standard practice. However, if you’re developing for just a touch based device, you can create some remarkable interactions that connect deeply with users.

this is a great time to have our jobs

(@skippykawakami)

touchy-feely ending. (magic.html)