



Sencha

Amazing Web Apps for Every Device

ExtJS 4.1 Performance

Nigel White, Software Architect

Examples & slides

- <https://github.com/ExtAnimal/SourceConf2012>
- <git@github.com:ExtAnimal/SourceConf2012.git>

Network Latency

CSS processing

Javascript execution

DOM

What slows your app down?

Network Latency

- Use Sencha SDK Tools
- <http://www.sencha.com/learn/getting-started-with-ext-js-4>
- Concatenate all Javascript into one file
- `sencha create jsb -a index.html -p app.jsb3`
- `sencha build -p app.jsb3 -d .`



CSS processing

- Avoid inefficient selectors
- Selectors are matched right to left
- `.HeaderContainer .nav span`



Javascript execution

- Avoid older and badly written JS engines. ;)
- Optimize code which is repeated.
- Optimize any code executed at render or initial layout time in an ExtJS app
- Better still, do not perform extra processing at render time.



Javascript execution

- Move invariant expressions out of loops
- Use `for (...) {}` rather than `Ext.Array.each`
- If a function only performs its functionality under some condition, test that condition **outside** the call.
- Setup and teardown of a call frame is slow on bad JS engines.



Array Grid							
Price ▾		Change					
	Get total in bad way		Price	Change	% Change	Last Updated	
	Get total in good way						
			\$71.72	0.02	0.03%	09/01/2012	⊖ ✓
			\$29.01	0.42	1.47%	09/01/2012	⊖ ✓
		Altria Group Inc	\$83.81	0.28	0.34%	09/01/2012	⊖ ✓
		American Express Company	\$52.55	0.01	0.02%	09/01/2012	⊖ ✓
		American International Group, Inc.	\$64.13	0.31	0.49%	09/01/2012	⊖ ✓
		AT&T Inc.	\$31.61	-0.48	-1.54%	09/01/2012	⊖ ⚠
		Boeing Co.	\$75.43	0.53	0.71%	09/01/2012	⊖ ✓
		Caterpillar Inc.	\$67.27	0.92	1.39%	09/01/2012	⊖ ✓
		Citigroup, Inc.	\$49.37	0.02	0.04%	09/01/2012	⊖ ✓
		E.I. du Pont de Nemours and Company	\$40.48	0.51	1.28%	09/01/2012	⊖ ✓
		Exxon Mobil Corp	\$68.10	-0.43	-0.64%	09/01/2012	⊖ ⚠
		General Electric Company	\$34.14	-0.08	-0.23%	09/01/2012	⊖ ⚠
		General Motors Corporation	\$30.27	1.09	3.74%	09/01/2012	⊖ ✓

A Menu of operations per column

Github

[examples/grid/array-grid.html](https://github.com/yourusername/examples/tree/master/grid/array-grid.html)

```
tbar: [{
    text: 'Price',
    field: 'price', // Button configured with column
    menu: [{
        text: 'Get total in bad way',
        handler: badTotalFn
    }, {
        text: 'Get total in good way',
        handler: goodTotalFn
    }]
}, {
    text: 'Change',
    menu: []
}]
```

Column's Button knows the field

```
function badTotalFn(menuItem) {  
    var r = store.getRange(),  
        total = 0;  
  
    Ext.Array.each(r, function(rec) {  
        total += rec.get(menuItem.up('button').field);  
    });  
}
```

What's wrong with this?

```
function goodTotalFn(menuItem) {  
    var r = store.getRange(),  
        field = menuItem.up('button').field;  
    total = 0;  
  
    for (var j = 0, l = r.length; j < l; j++) {  
        total += r[j].get(field);  
    }  
}
```

Why is this better?

Browser	Bad	Good
Chrome	1700ms	10ms
IE9	18000ms	500ms
IE6	Gave up	532ms

The results (10000 iterations)

Use PageAnalyzer

- [extjs/examples/page-analyzer/page-analyzer.html](http://extjs.com/examples/page-analyzer/page-analyzer.html)
- Analyzes method calls of selected classes
- Start Chrome with `--enable-benchmarking` for microsecond time measurement accuracy.



```
{  
  "Component.up": {  
    "Ext.Component": "up"  
  },  
  "CQ.is": {  
    "Ext.ComponentQuery": "!is"  
  }  
}
```

Performance Accumulators

Name	Time	Time/Call	Samples	Min Calls ▲	Max Calls
[-] Path: /~nigewhite/SourceConf2012/extjs/examples/grid/array-grid.html (x - 1)					
BeforeReady	227	227	1	1	1
Initialize	618	618	1	1	1
Component.up	0	0	1	104	104
CQ.is	2	0.01	1	220	220
[-] Path: /~nigewhite/SourceConf2012/extjs/examples/grid/array-grid.html (x - 2)					
BeforeReady	219	219	1	1	1
Initialize	604	604	1	1	1
Component.up	0	0	1	76	76
CQ.is	0	0	1	164	164

Results (bad way first)

Reduce DOM burden

- Reduce Container nesting
- Use the simplest Container or Layout which does the job



```
{
  xtype: 'tabpanel',
  items: [{
    title: 'Results',
    items: {
      xtype: 'grid'
      ...
    }
  }]
}
```

What's wrong with this?

```
{  
    xtype: 'tabpanel',  
    items: [{  
        title: 'Results',  
        xtype: 'grid',  
        ...  
    }]  
}
```

Why is this better?

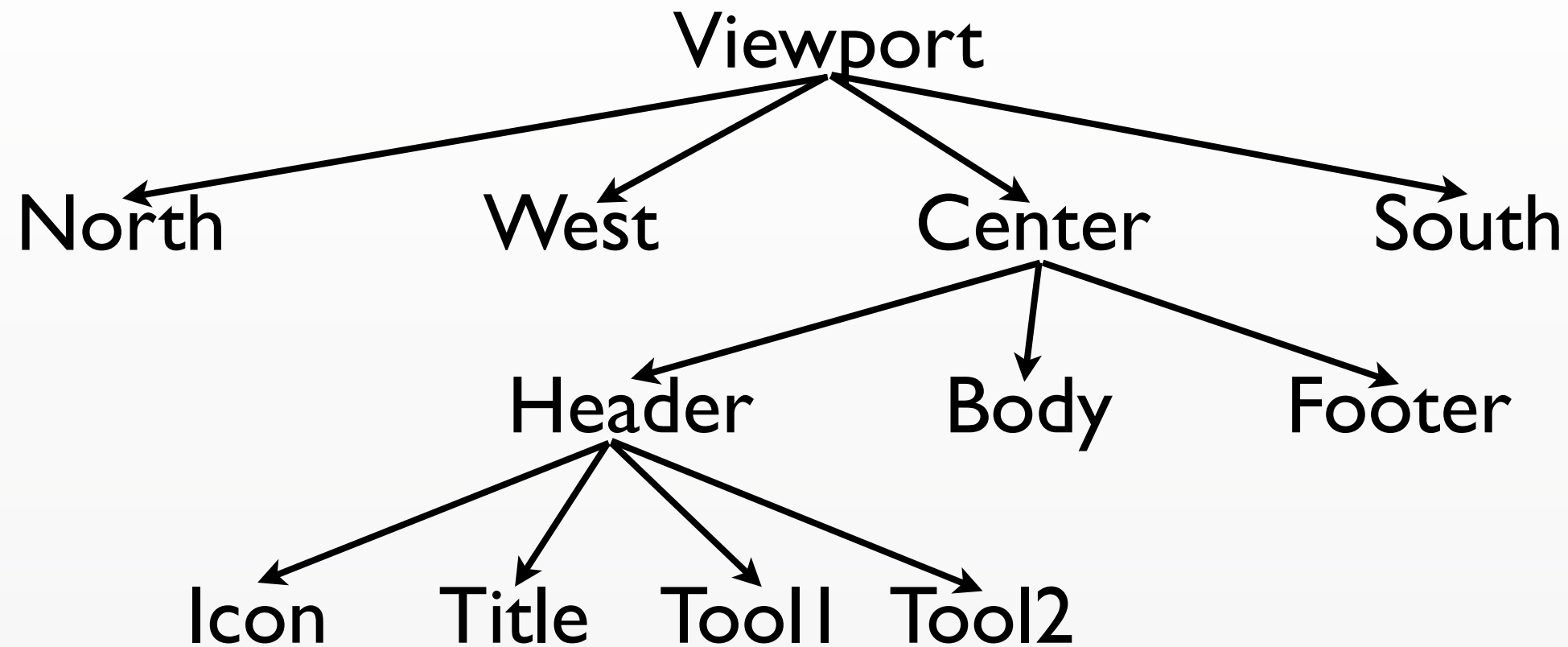


Why is this important?

- In ExtJS 4.x, more widgets are Containers which must layout out child Components using a Layout Manager. eg: Header, TabBar.
- In ExtJS 4.x, All Components lay out their internal DOM structure using a Component Layout Manager (Where 3.x simply had an onResize method)



The Component Tree



- Walked twice, visiting **every** node.
- beforeRender & getRenderTree: Finished pre-render configs, Contributes a DomHelper config which is converted to markup and added into the buffer.
- onRender & afterRender: Inform the Component about its element. Component grabs child element references (childEls)



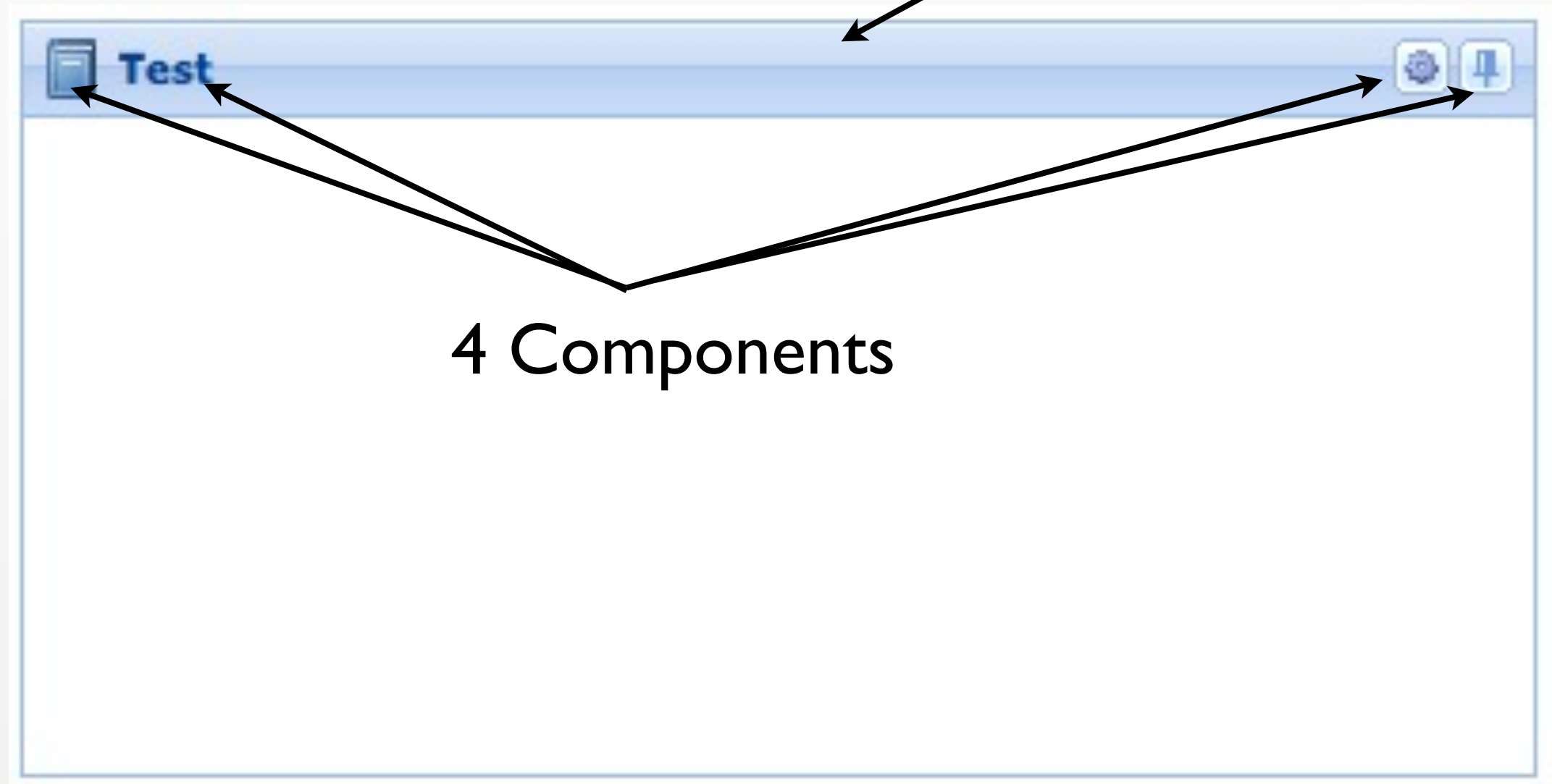
Example Panel

```
Ext.create('Ext.panel.Panel', {  
    width: 400, height: 200,  
    icon: '../shared/icons/fam/book.png',  
    title: 'Test',  
    tools: [{  
        type: 'gear'  
    }, {  
        type: 'pin'  
    }],  
    renderTo: document.body  
});
```



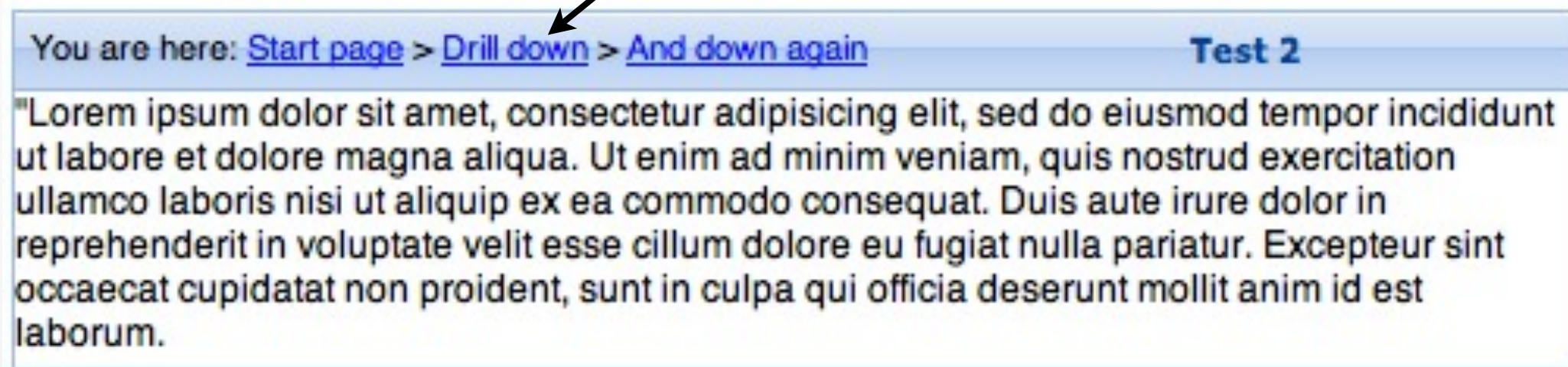
Example Panel

Header = Container + HBox



More flexibility than ExtJS 3.x

Components in Headers:
“Breadcrumbs” DataView



“shrinkwrap” sizing where content drives height.
(and/or width for non-textual content)



Github

- `examples/panel/panel-test1.html`
- `examples/panel/panel-test2.htm`

Layout analysis, auto sized.

Layouts					
Layout	Box Parent	Time	Calls	Avg Time	Tot Time
▷ Run #9 (2012-04-23 16:1:		15	1	15	15
▷ Run #10 (2012-04-23 16:1:		47	1	47	47

- Two layout runs due to `deferInitialRefresh` default as `true`
- More expensive layout after the data is refreshed
- The heights of all Components in header must be read from the DOM
- That requires that the width be set – because width **may** affect height of Components (wrapping)
- Only then can the Panel's body height be determined (and therefore any dock: 'bottom' Components positioned)



Layout analysis, fixed size

Layouts					
Layout	Box Parent	Time	Calls	Avg Time	Tot Time
▶ Run #6 (2012-04-23 16:10)		15	1	15	15

- One layout run.
- Faster because the height was known before any layout was performed, so docking could proceed immediately.



More tries to lay out

Layout	Time	Calls	Avg Time	Tot Time
Run #1 (2012-04-27 12:16:16)	16.48	1	16.48	16.48
Flush	1.8	7	0.26	1.8
Invalidate	5.59	1	5.59	5.59
CompleteLayout	0.2	2	0.1	0.2
FinalizeLayout	0.01	1	0.01	0.01
FinishedLayout	0.31	1	0.31	0.31
AfterLayout	0.53	1	0.53	0.53
panel<dock>	2.21	3	0.74	7.29
panel<autocontainer>	0.34	1	0.34	0.34

It took **3** tries to do the dock layout.
Because it has to be done in stages.



Waves of change

- Write to DOM then read from DOM == a browser reflow.
Do as few as possible.
- Layouts all executed - *publish* all the information they can using what is already *published* by other layouts.
- If no layout has made progress, then all *published* values are *flushed* to the DOM.
- Layouts begin another cycle - they may read the results from that flush.

“publish” means the value is available to other layouts.
These values are *not* yet in the DOM



First try

- Width is already *published* - configured width.
- Not height - Panel is shrinkwrapping content.
- Try to dock the header - *publish* its width.
- *Publish* body x position and width.
- Header height not known, (internal components have not performed layouts and published sizes) so no further progress can be made on the dock.



Second try

- Header has published its height.
- We can therefore position the body - $y = 25\text{px}$
- But the body height cannot be measured because no flush has taken place to set the body element width.
- No further progress can be made on finishing the dock layout with the total panel height

First DOM flush takes place



Third try

- Body's width has now been flushed to the DOM, so its height can be measured.
- Header already published its height.
- The total height of the Panel can now be calculated
- Done!



TL;DL

- Avoid “auto” sizing (shrinkwrapping) in the initial page layout. It will require multiple passes to flush the sizes, then measure.
- Avoid size constraints (minHeight, maxWidth etc). If they kick in, they will cause another round of layout calculations.
- Box layout’s stretchmax config causes multiple layout attempts.

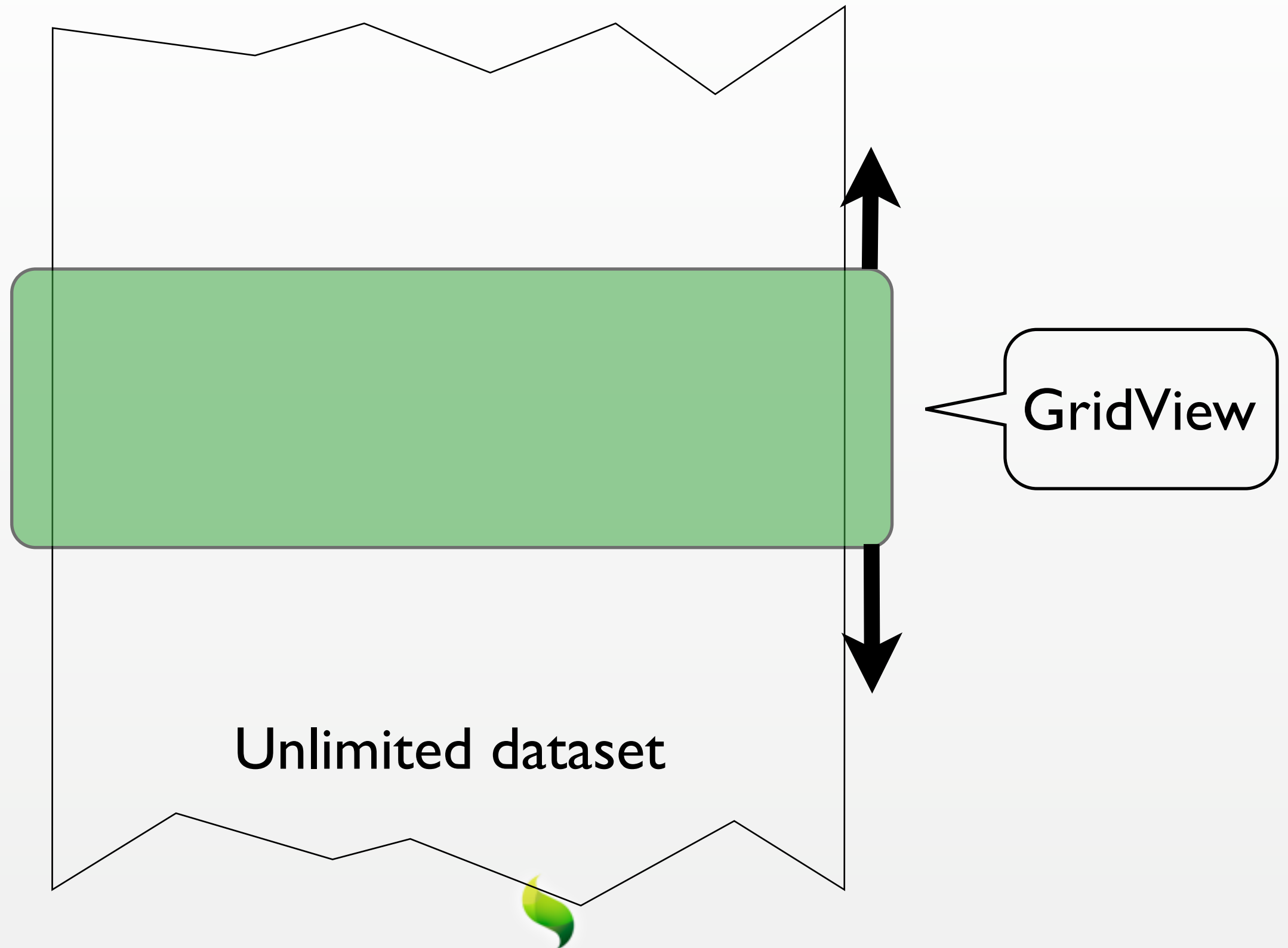


Grid Performance

- Table size affects performance
- Column count has a large effect.
- Avoid `trackMouseOver` on slow browsers
- Where dataset is large, use *buffered rendering*.



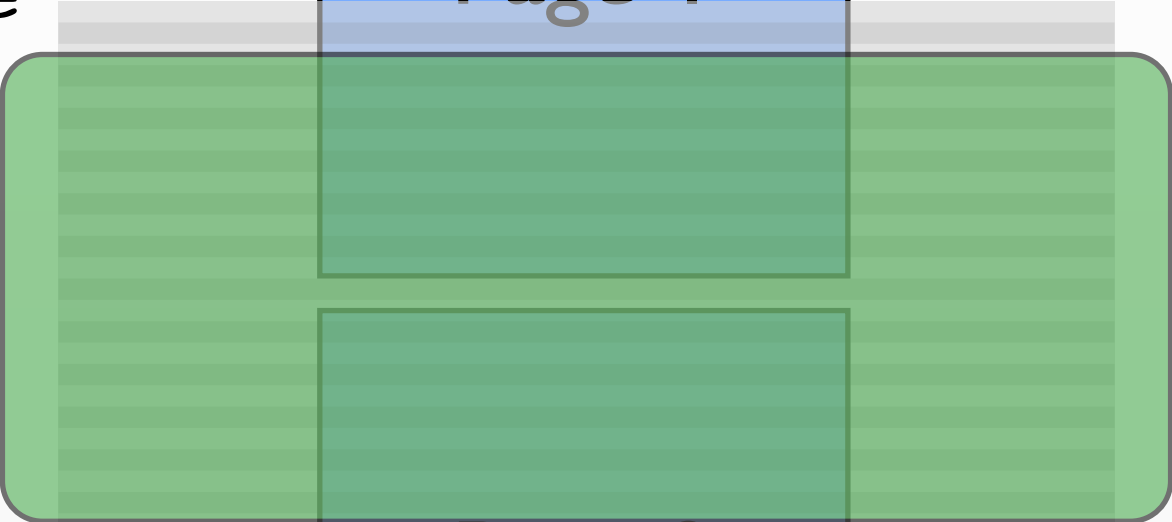
Letterbox view



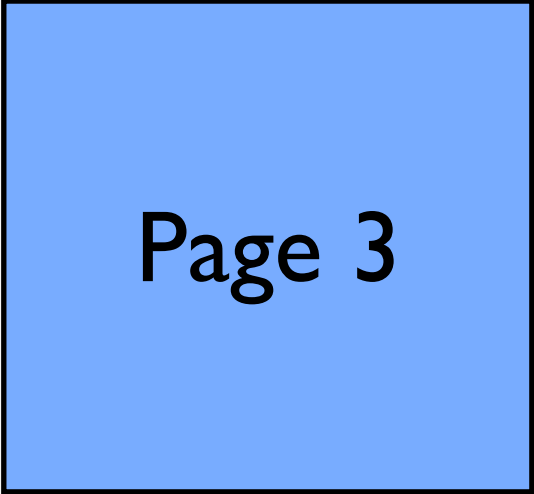
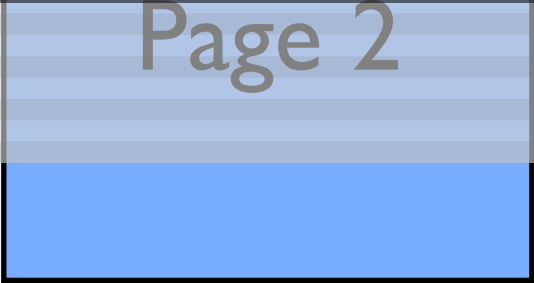
PagingScroller
trailingBufferZone



Store
trailingBufferZone



PagingScroller
leadingBufferZone



Store
leadingBufferZone



Questions?

- <https://github.com/ExtAnimal/SourceConf2012>
- <git@github.com:ExtAnimal/SourceConf2012.git>

