FAST and EFFICIENT
Tizen HTML5 mobile applications
@akisaarinen
Reaktor
FAST & EFFICIENT
Tizen™ is a mobile platform
1 Measure
2 Start-up time
3 Run-time performance
1 MEASURE

(1) Measure
(2) Start-up time
(3) Run-time
Measure before optimizing
Available tools

• WebKit Web Inspector
• Tizendev: start-up
• Tizendev: framerate

(1) Measure  (2) Start-up time  (3) Run-time
WebKit Web Inspector

(1) Measure
(2) Start-up time
(3) Run-time
Tizendev

http://github.com/reaktor/tizendev

- Automated deploying of app
- Automated start-up timing
- Automated FPS measurements
tizendev: start-up time

runs: 30
mean: 1708ms
std: 63ms
<table>
<thead>
<tr>
<th></th>
<th>Measure</th>
<th>Start-up time</th>
<th>Run-time</th>
</tr>
</thead>
<tbody>
<tr>
<td>samples:</td>
<td>100</td>
<td>mean: 58 FPS</td>
<td>std: 4 FPS</td>
</tr>
</tbody>
</table>
Available tools

- WebKit Web Inspector
- Tizendev: start-up
- Tizendev: framerate
2 START-UP TIME

(1) Measure
(2) Start-up time
(3) Run-time
Less is more

IT'S SO OBVIOUS!!!
• Lazy-loading
• Minification
• Reflow
• Native API calls
• Parallelization

(1) Measure
(2) Start-up time
(3) Run-time
Monolithic app

Large codebase, all loaded and parsed at start-up time
Monolithic app

Large codebase, all loaded and parsed at start-up time

(1) Measure
(2) Start-up time
(3) Run-time
Lazy-loading

Code to show first screen

Modularized pieces to show other views on-demand
Minify: UglifyJS

(1) Measure

(2) Start-up time

(3) Run-time
1 kilobyte \approx 1 \text{ ms}
Avoid reflow
Affects also run-time
Example:
Calling width() of an element
container.find("li").each(function() {
    var listItem = $(this);
    listItem.text(item.width());
});

(1) Measure

(2) Start-up time

(3) Run-time

(1) Measure
(2) Start-up time
(3) Run-time

container.find("li").each(function() {
  var listItem = $(this);
  listItem.text(item.width());
});
container.detach();

container.find("li").each(function()
    
    var listItem = $(this);
    
    listItem.text(item.width());

});

container.appendTo($("body"));
container.detach();

container.find("li").each(function() {
  var listItem = $(this);
  listItem.text(item.width());
});

container.appendTo($("body"));
1000 elements (MacBook Pro)

2000 ms → 60 ms
Native APIs

- Defer execution
- Use localstorage
- Only fetch needed data
Parallelize

- Resources
- Service calls
Parallelize

(1) Measure
(2) Start-up time
(3) Run-time
Parallelize

Could be parallelized?

(1) Measure
(2) Start-up time
(3) Run-time
Parallelize

(1) Measure
(2) Start-up time
(3) Run-time

~150 ms
• Do lazy-loading
• Use minification
• Avoid reflow
• Careful with native APIs
• Parallelize

(1) Measure  (2) Start-up time  (3) Run-time
3 RUN-TIME PERFORMANCE

(1) Measure
(2) Start-up time
(3) Run-time
60 FPS
• DOM modifications
• Pre-loading
• CSS3 transitions
• Scrolling
DOM = SLOW
display: none;

+ 5-10 FPS
Pre-loading images/views

1 (pre-load)

2 visible

3 (pre-load)

(1) Measure  (2) Start-up time  (3) Run-time
Accelerated CSS3 transitions

(1) Measure
(2) Start-up time
(3) Run-time
NO: jQuery.animate()

YES: CSS3

(1) Measure  (2) Start-up time  (3) Run-time
NO: left: 0px -> 100px

YES: translate3d()
**NO:** background-color: ...;

**YES:** opacity: 0.2;
Enable 3D acceleration

-webkit-transform: translate3d(0,0,0);

http://stackoverflow.com/questions/3461441/prevent-flicker-on-webkit-transition-of-webkit-transform
Trigger animation in next render cycle

(1) Measure
(2) Start-up time
(3) Run-time
setTimeout(function() {
    element.css(
        "-webkit-transform",
        "translate3d(100,0,0)"
    );
}, 0);
Momentum scrolling

**NO:** iScroll or other JavaScript library

**NO:** overflow: scroll;

**YES:** -webkit-overflow-scroll: touch;
• DOM is slow
• Do pre-loading
• Use CSS3 transitions
• Use overflow scrolling
Summary

1. Measure
2. Start-up time
3. Run-time performance
Summary

• Performance is important
• Measure before optimizing
• Minimize actions at start-up
• Pay attention to FPS
Thank you!

@akisaarinen
Reaktor