FAST and EFFICIENT
Tizen HTML5 mobile applications

@akisaarinen

Reaktor

FAST & EFFICIENT

TIZEN™

is a mobile platform

TIZEN™

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

MEASURE

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

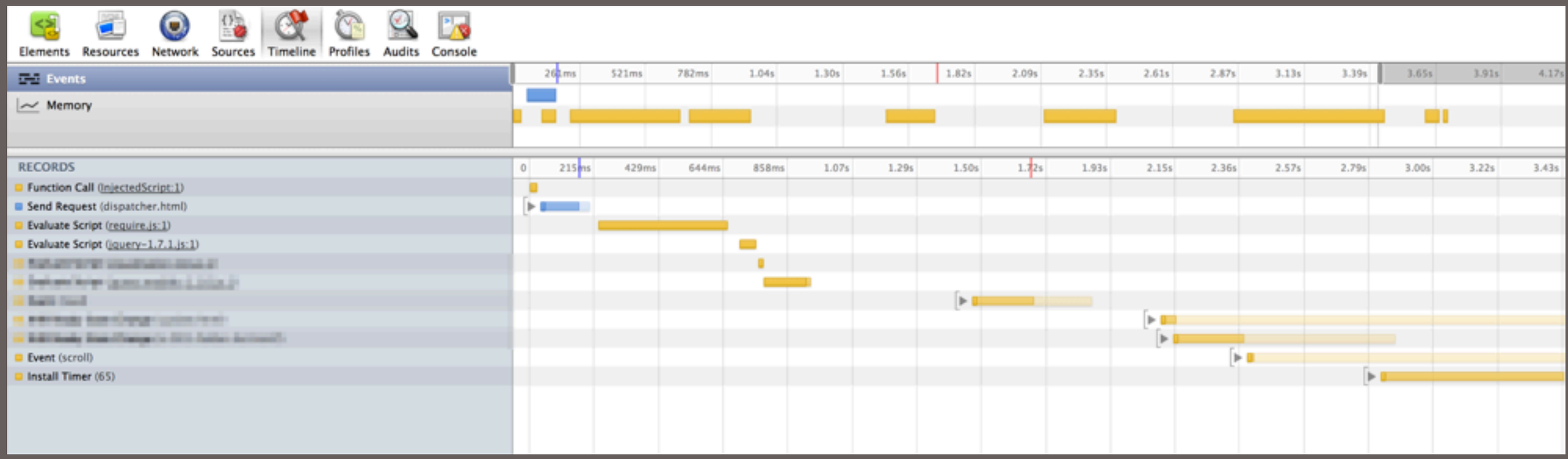
Measure **before** optimizing

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

Available tools

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

WebKit Web Inspector



(1) Measure

(2) Start-up time

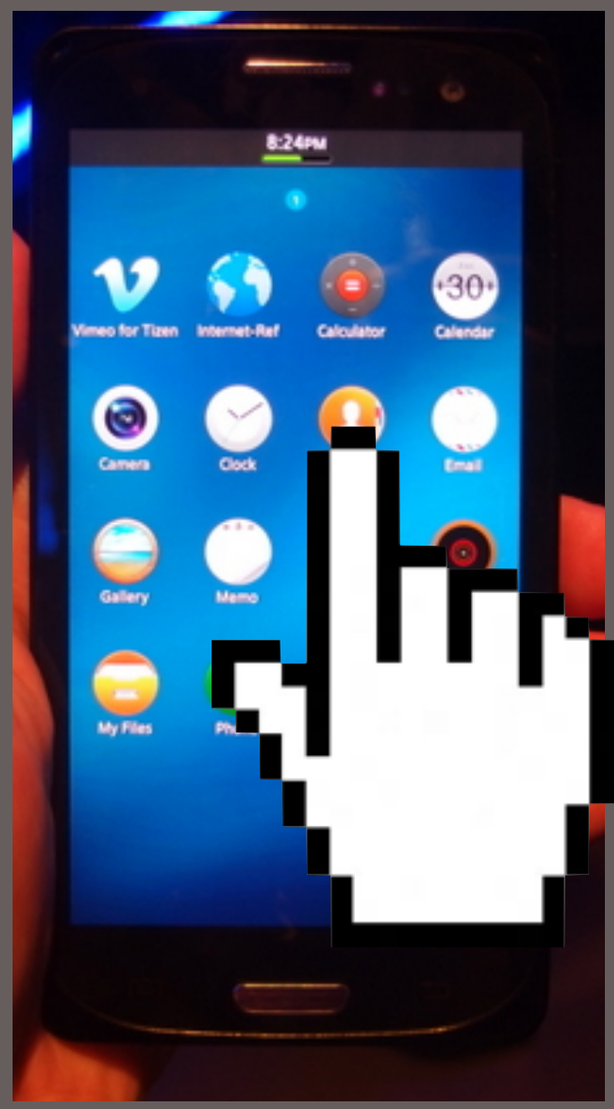
(3) Run-time

Tizendev

http://github.com/reaktor/tizendev

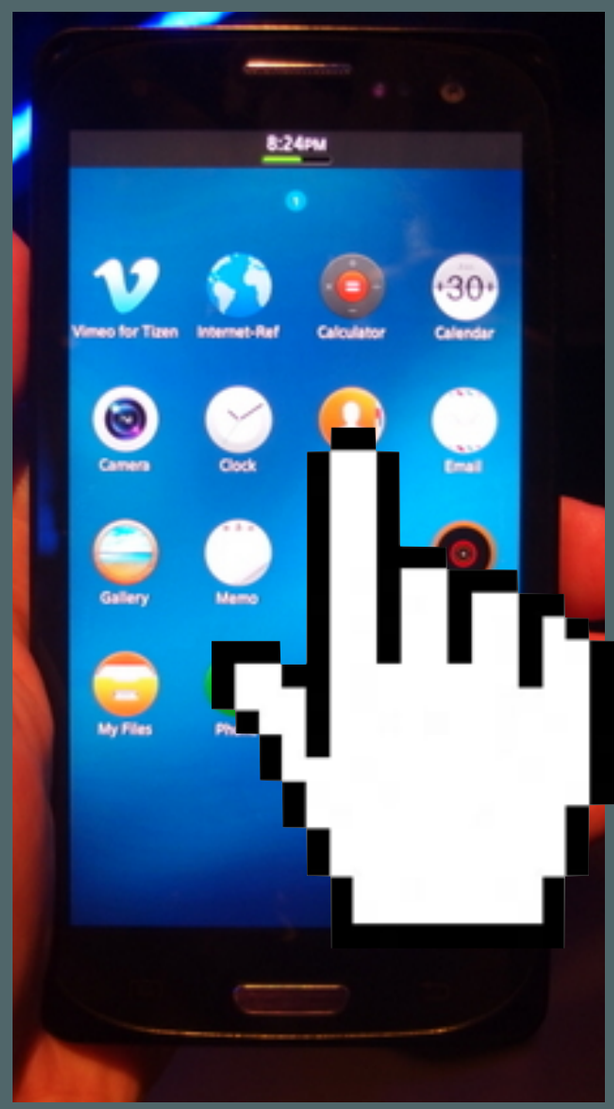
- Automated deploying of app
- Automated start-up timing
- Automated FPS measurements

tizendev: start-up time




```
runs:     30
mean: 1708ms
std:    63ms
```

tizendev: framerate




```
samples:   100
mean:   58 FPS
std:     4 FPS
```

Available tools

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

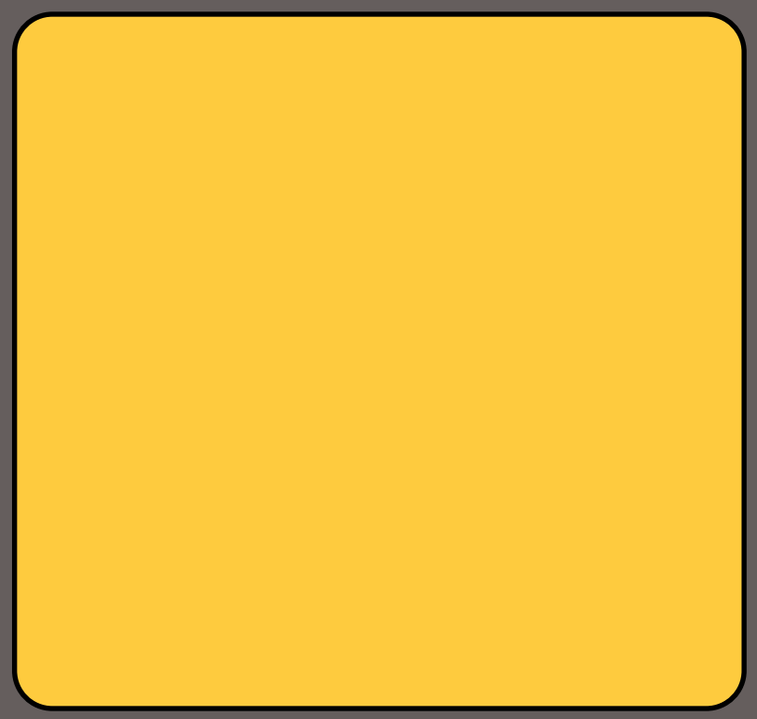
2 START-UP TIME

Less is more

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

- Lazy-loading
- Minification
- Reflow
- Native API calls
- Parallelization

Monolithic app

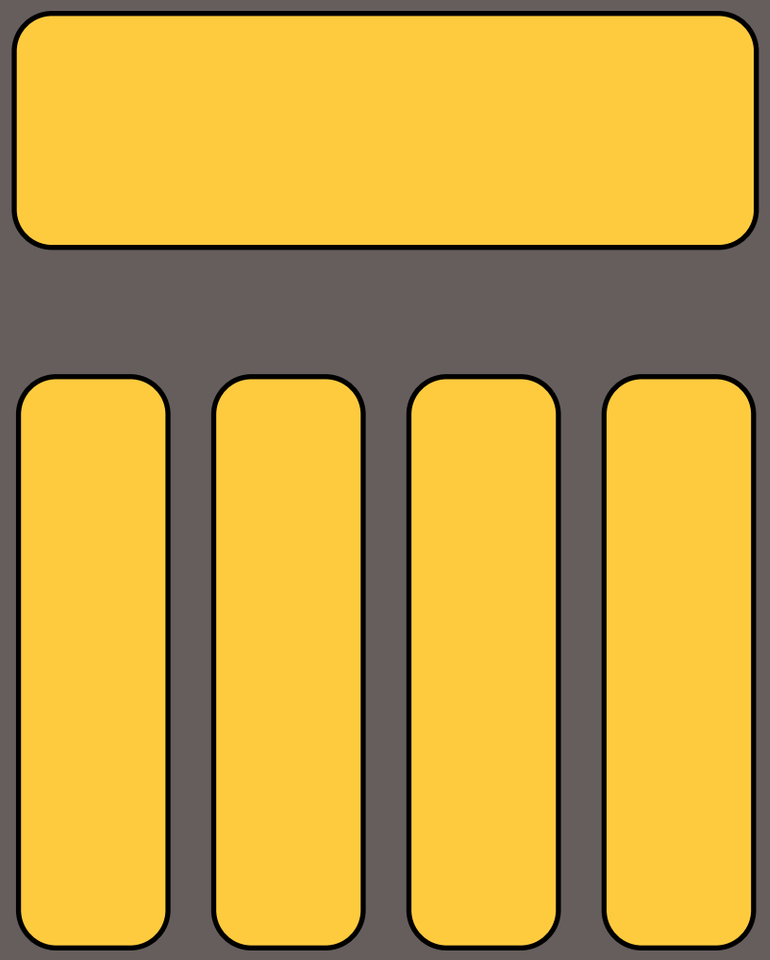


Large codebase,
all loaded and parsed
at start-up time

Monolithic app

Large codebase,
all loaded and parsed
at start-up 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 ~= 1 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());
});
```

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

forces reflow

```
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"));
```

prevents reflow

1000 elements (MacBook Pro)

2000 ms → 60 ms

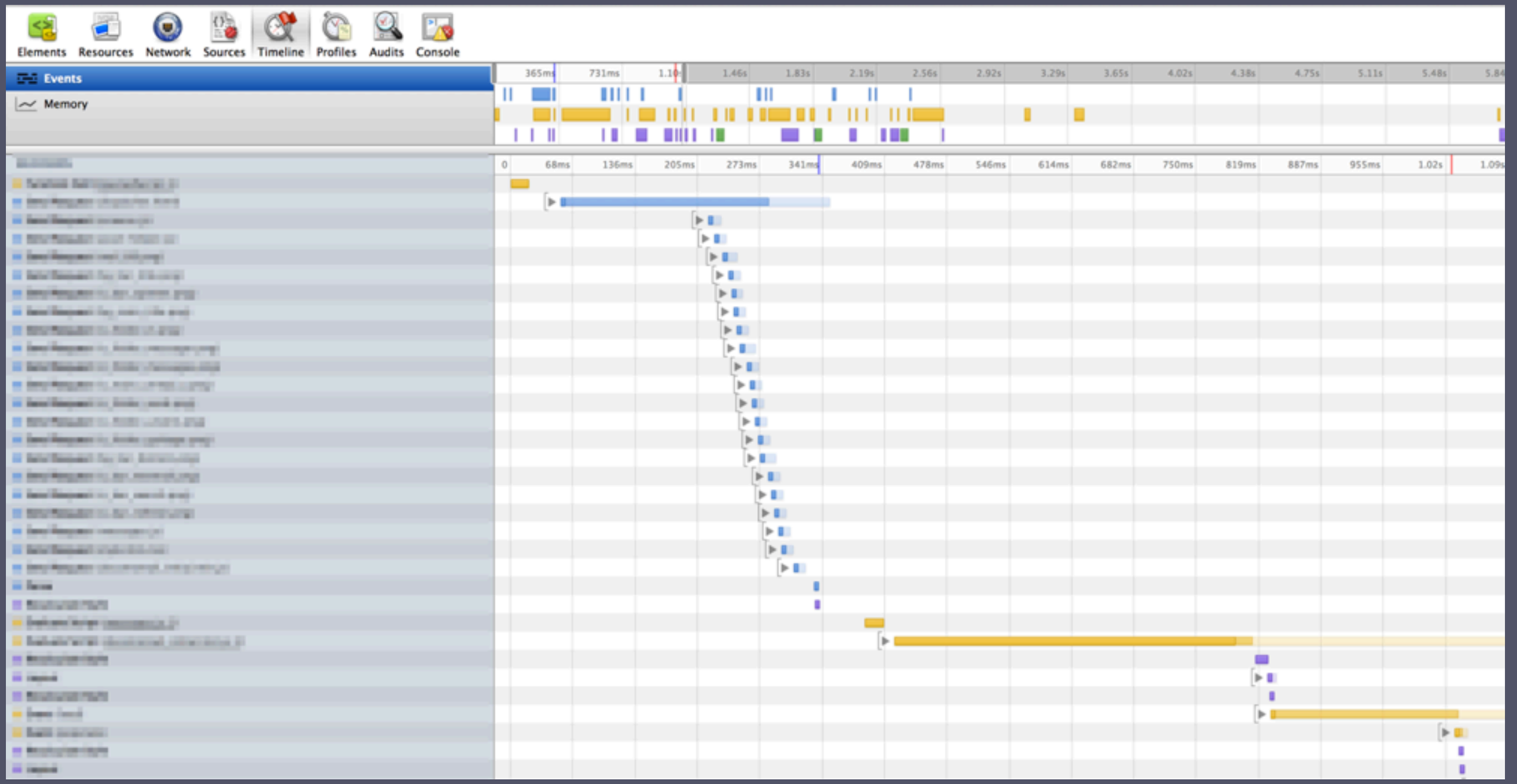
Native APIs

- Defer execution
- Use localstorage
- Only fetch needed data

Parallelize

- Resources
- Service calls

Parallelize

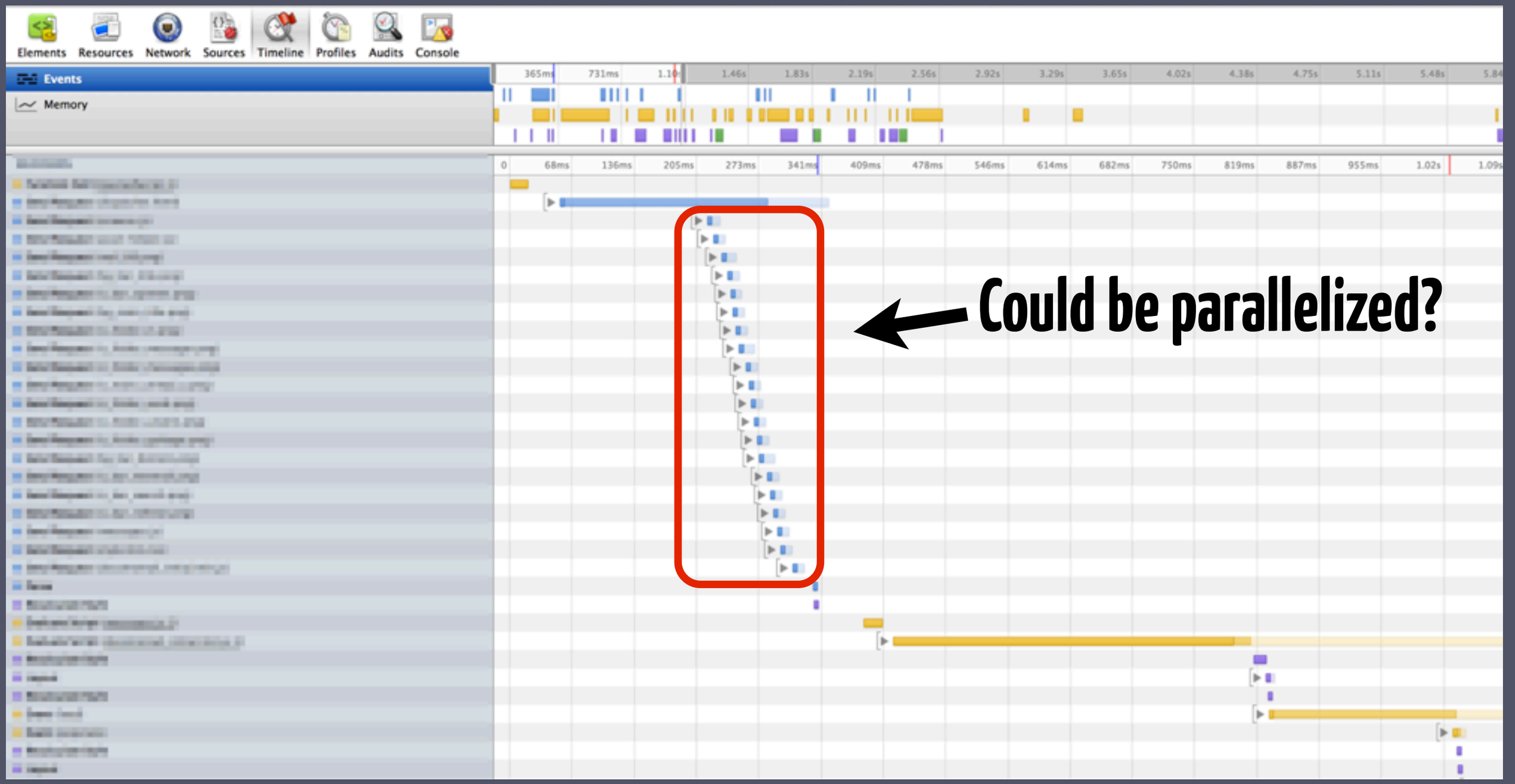


(1) Measure

(2) Start-up time

(3) Run-time

Parallelize

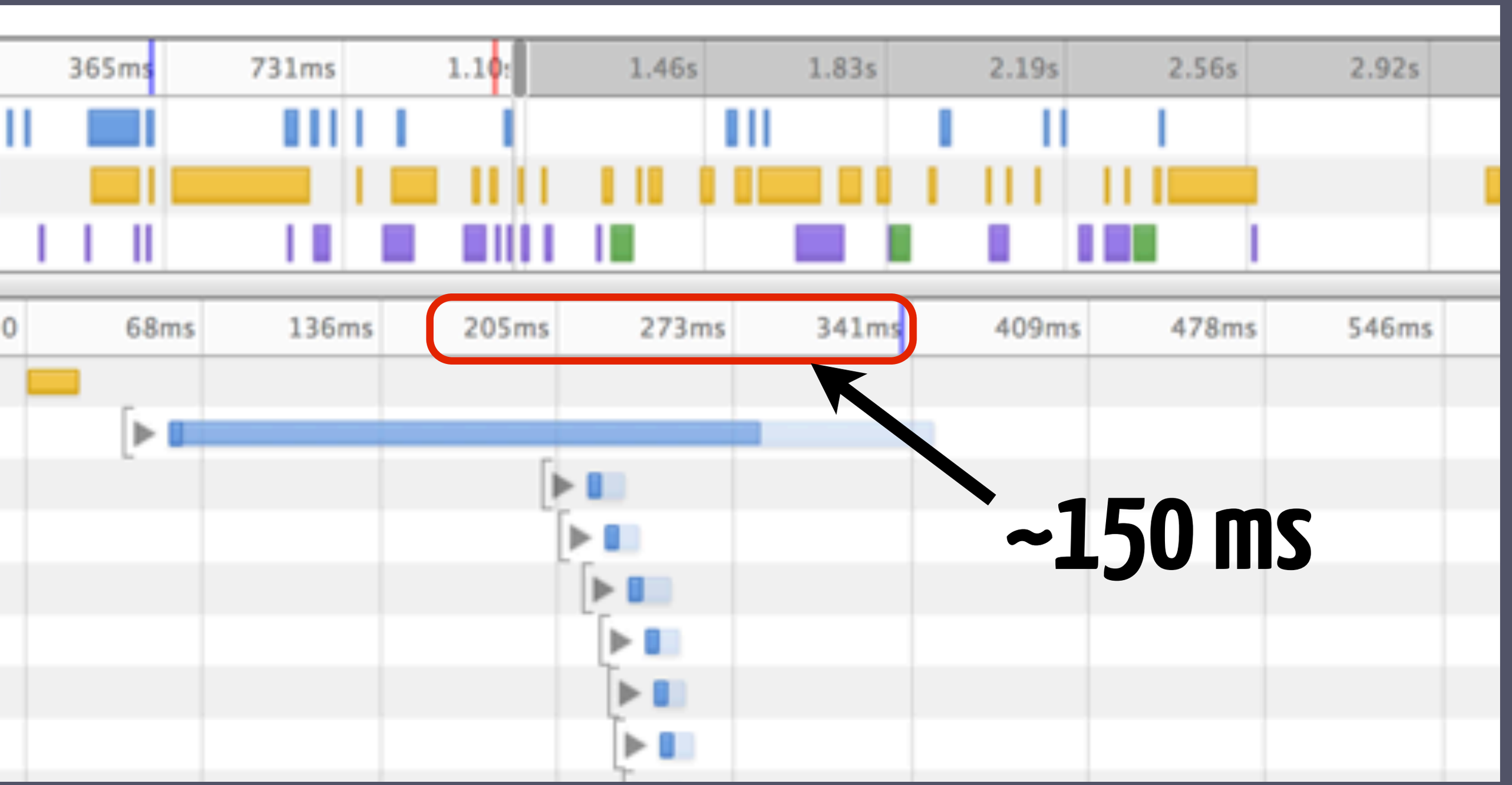


(1) Measure

(2) Start-up time

(3) Run-time

Parallelize



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

- Do lazy-loading
- Use minification
- Avoid reflow

- Careful with native APIs
- Parallelize

3 RUN-TIME PERFORMANCE

60 FPS

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

- DOM modifications
- Pre-loading
- CSS3 transitions
- Scrolling

DOM

=

SLOW

(1) Measure

(2) Start-up time

(3) Run-time

display: none;

+ 5-10 FPS

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

Pre-loading images/views

1	**2**	3
(pre-load)	visible	(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

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

```
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