STEALING CHROMIUM: EMBEDDING HTML5 WITH THE SERVO BROWSER ENGINE

Lars Bergstrom
Mozilla Research

Mike Blumenkrantz
Samsung R&D America

Why a new web engine?

- Support new types of applications and new devices
- All modern browser engines (Safari, Firefox, Chrome) originally designed pre-2000
  - Coarse parallelism
  - Tightly coupled components
- Vast majority of security issues are related to the C++ memory model

Servo

- Written in a memory-safe systems language, Rust
- Architected for parallelism
  - Coarse (per-tab), as in Chrome
  - Lightweight (intra-page), too
- Designed for embedding

mozilla SAMSUNG

Rust - safe systems programming

- C++ syntax and idioms
- C++-like performance
- Memory safety
- Concurrency
- Parallelism

http://www.rust-lang.org

mozilla SAMSUNG

Familiar syntax and performance

```
1  fn main() {
2      let vec = [1i , 2 , 3];
3
4      for v in vec.iter() {
5          println!("{}", *v);
6      }
7  }
```

```
1
2
3
Program ended.
```

Memory safety without overhead

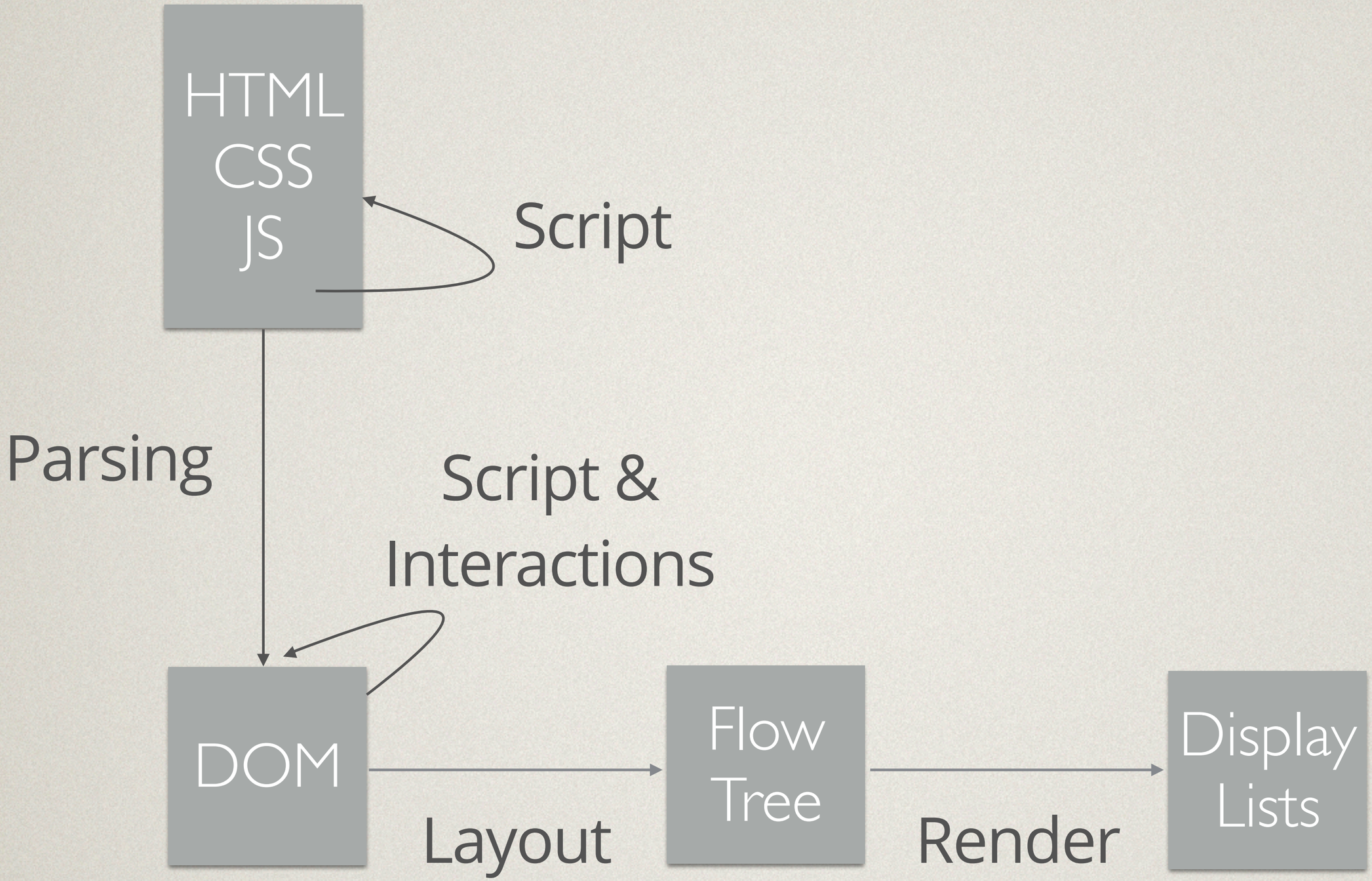
- Lifetimes and ownership ensure memory safety
  - No garbage collection
  - No reference counting
  - No C++ “smart” pointer classes

Example of code you can't write

```
1  fn main() {
2      let mut vec = vec!(1i , 2 , 3);
3      let mut vec2 = vec;
4      vec.push(3);
5  }
```

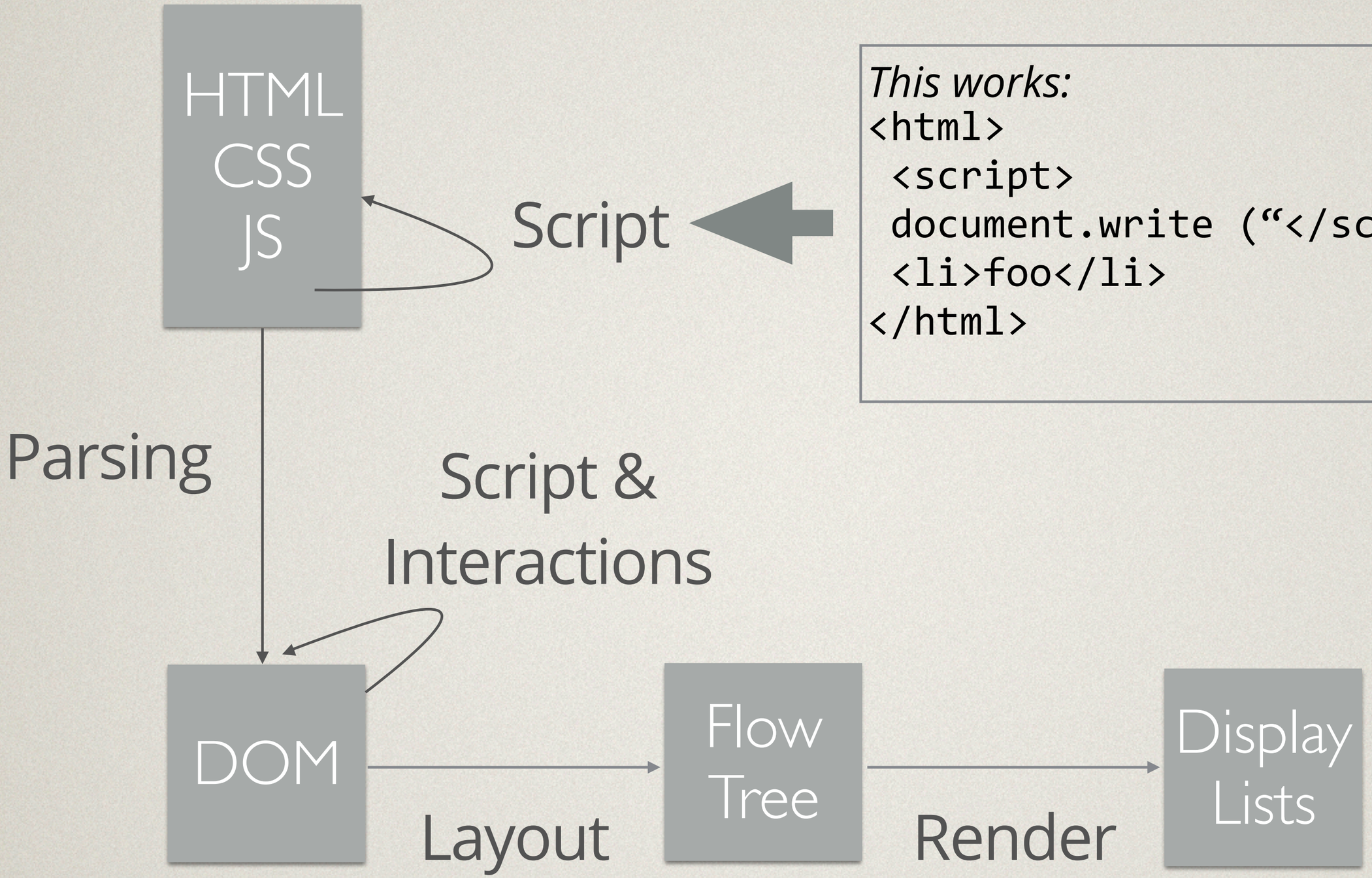
```
<anon>:4:5: 4:8 error: use of moved value: `vec`
<anon>:4     vec.push(3);
             ^~~
<anon>:3:9: 3:17 note: `vec` moved here because it has type `collections::vec::Vec<i
<anon>:3     let mut vec2 = vec;
                 ^~~~~~~~
error: aborting due to previous error
```

How a browser works



More details: http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

How a browser works



This works:

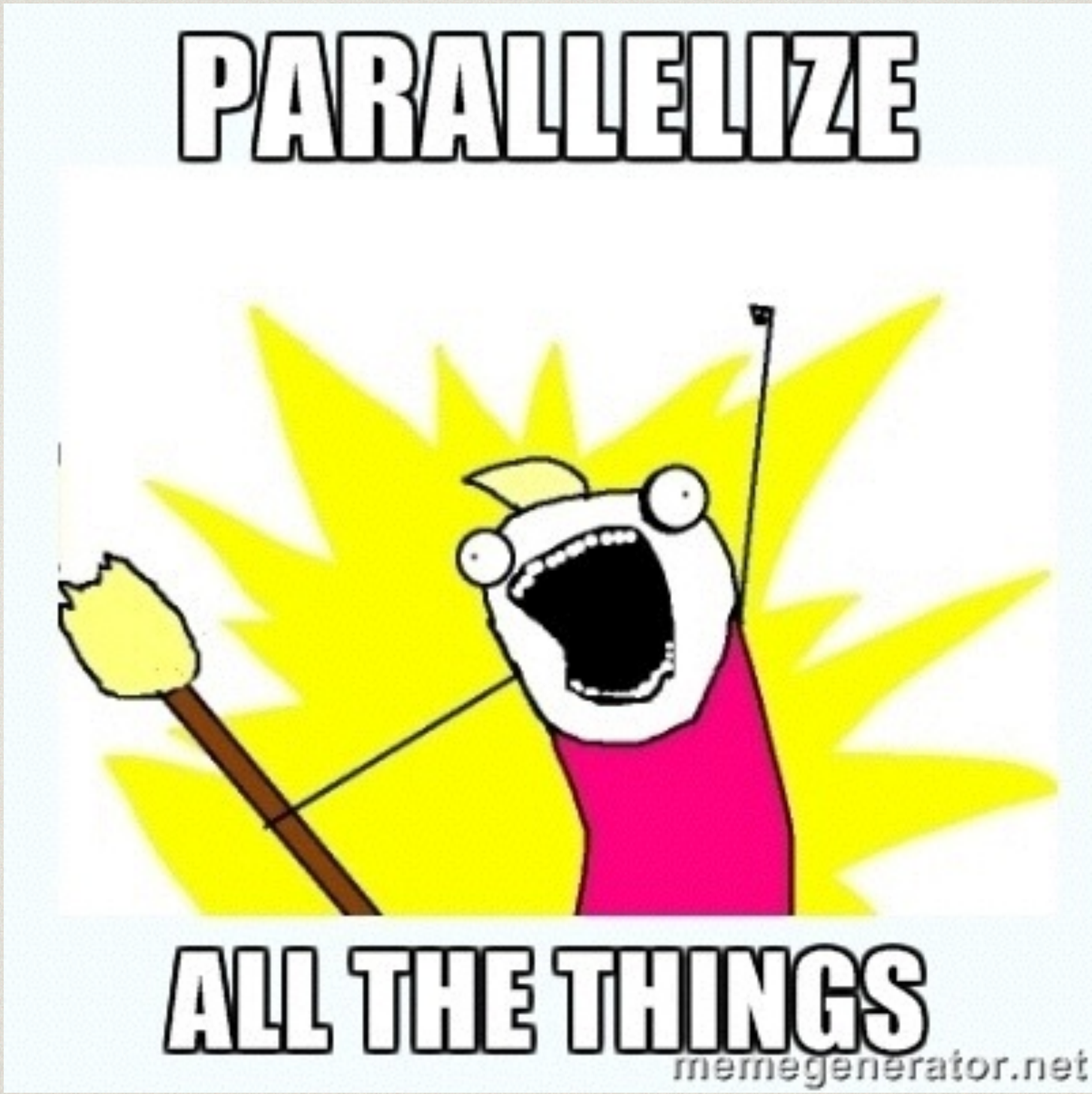
```
<html>
 <script>
 document.write (“</script>
 <li>foo</li>
</html>
```

More details: http://www.html5rocks.com/en/tutorials/internals/howbrowserswork/

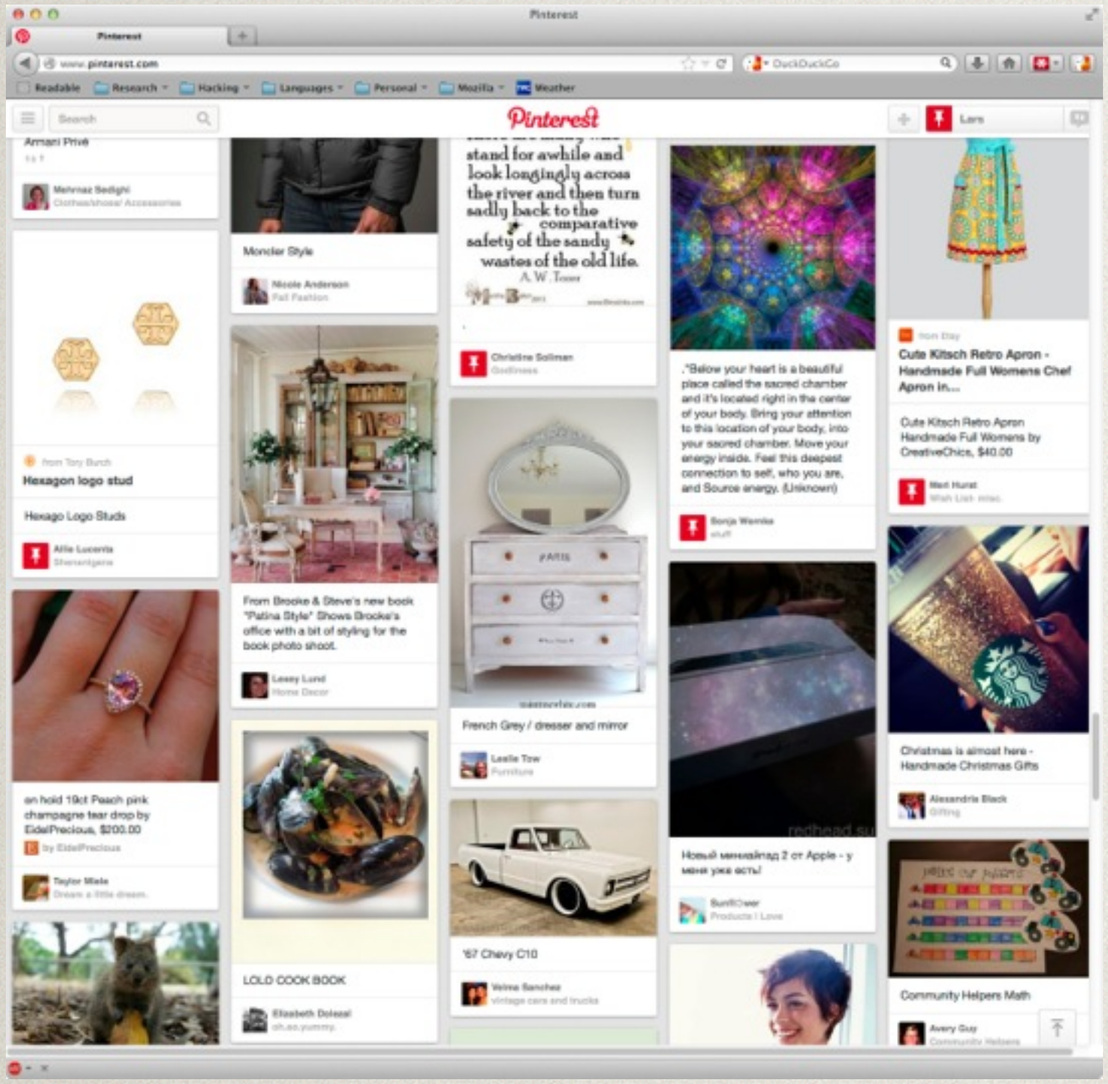
Timing breakdown

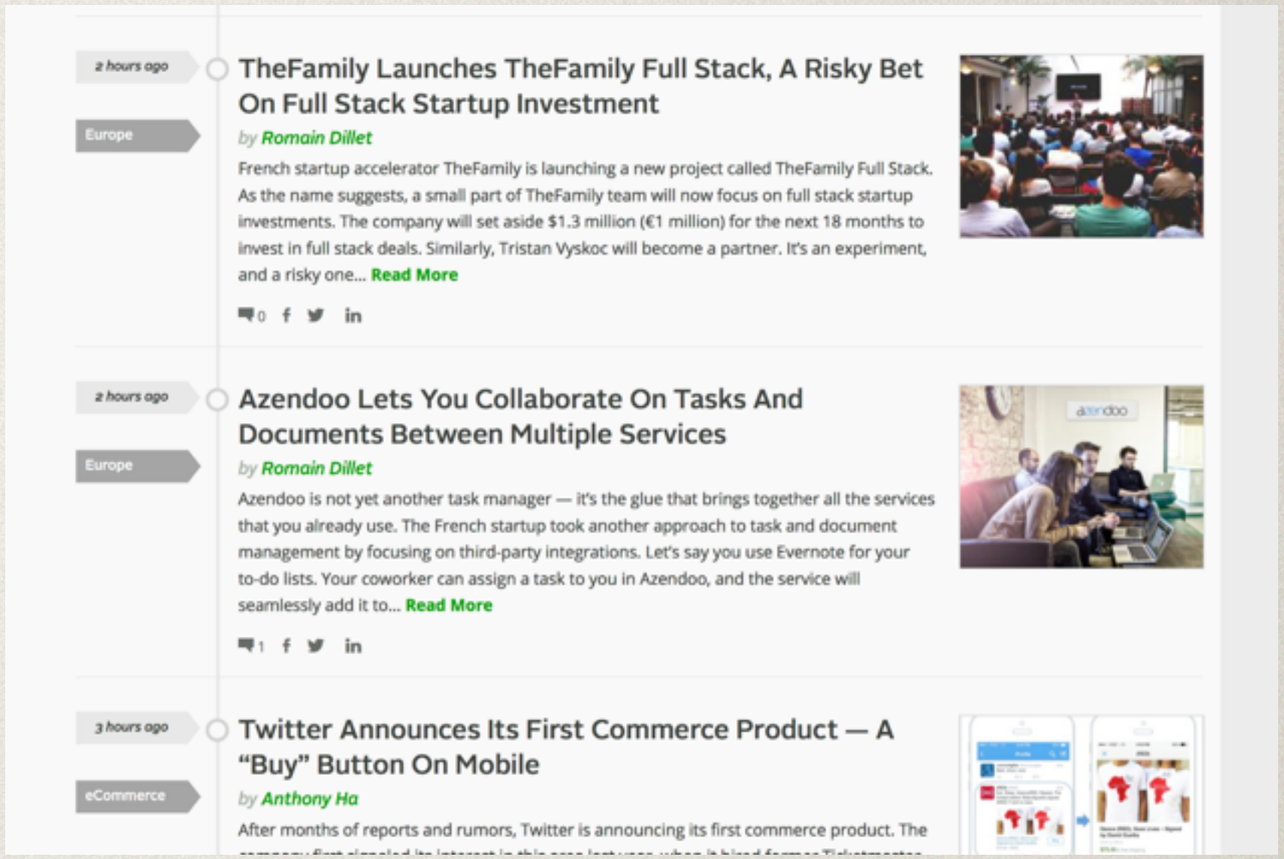
Task	Percentage
Runtime libraries	25%
Layout	22%
Windowing	17%
Script	16%
Painting to screen	10%
CSS styling	4%
Other	6%

Data from A Case for Parallelizing Web Pages. Mai, Tang, et. al. HOTPAR '12

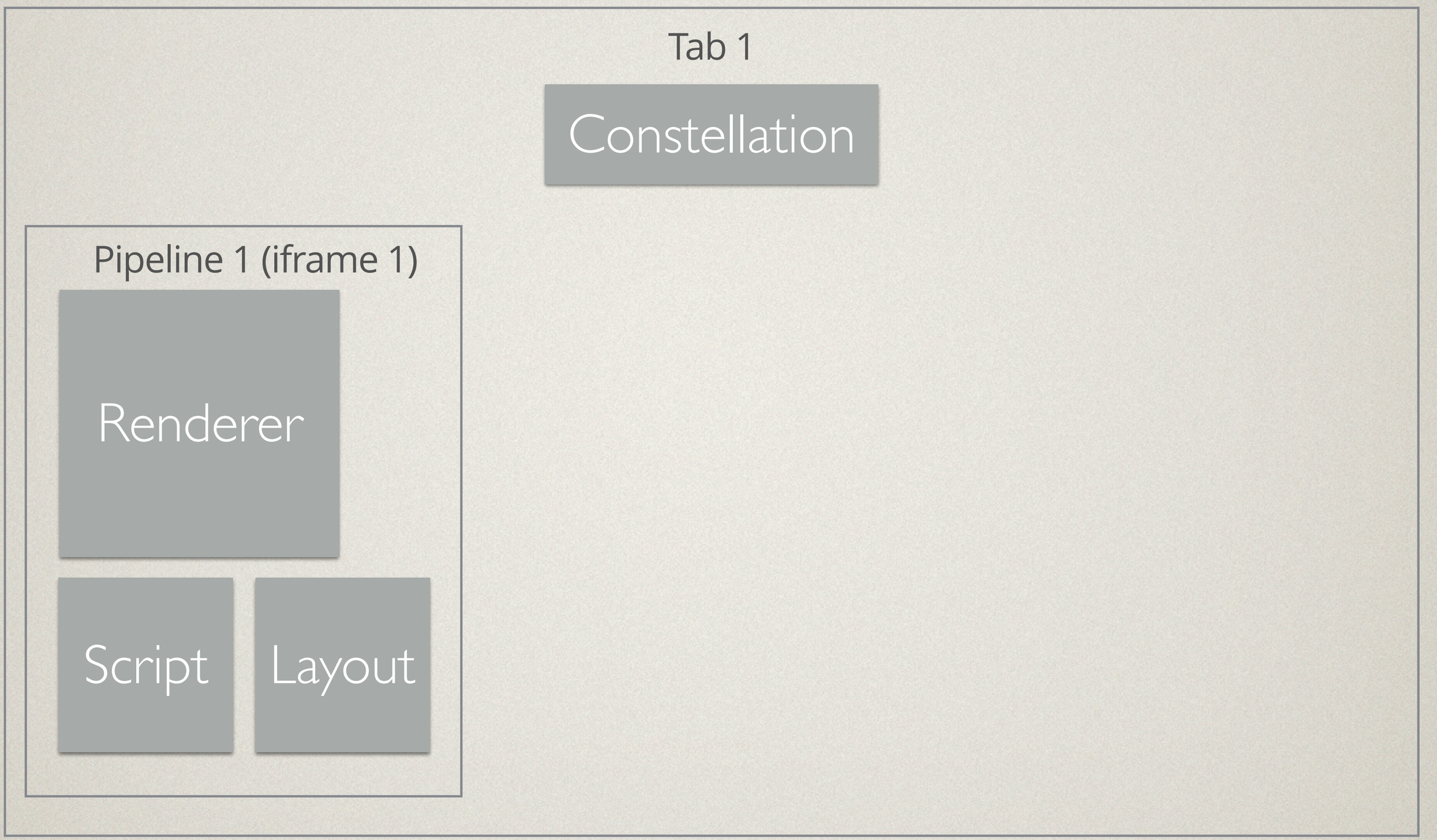

PARALLELIZE
ALL THE THINGS
memegenerator.net

Parallelism within pages

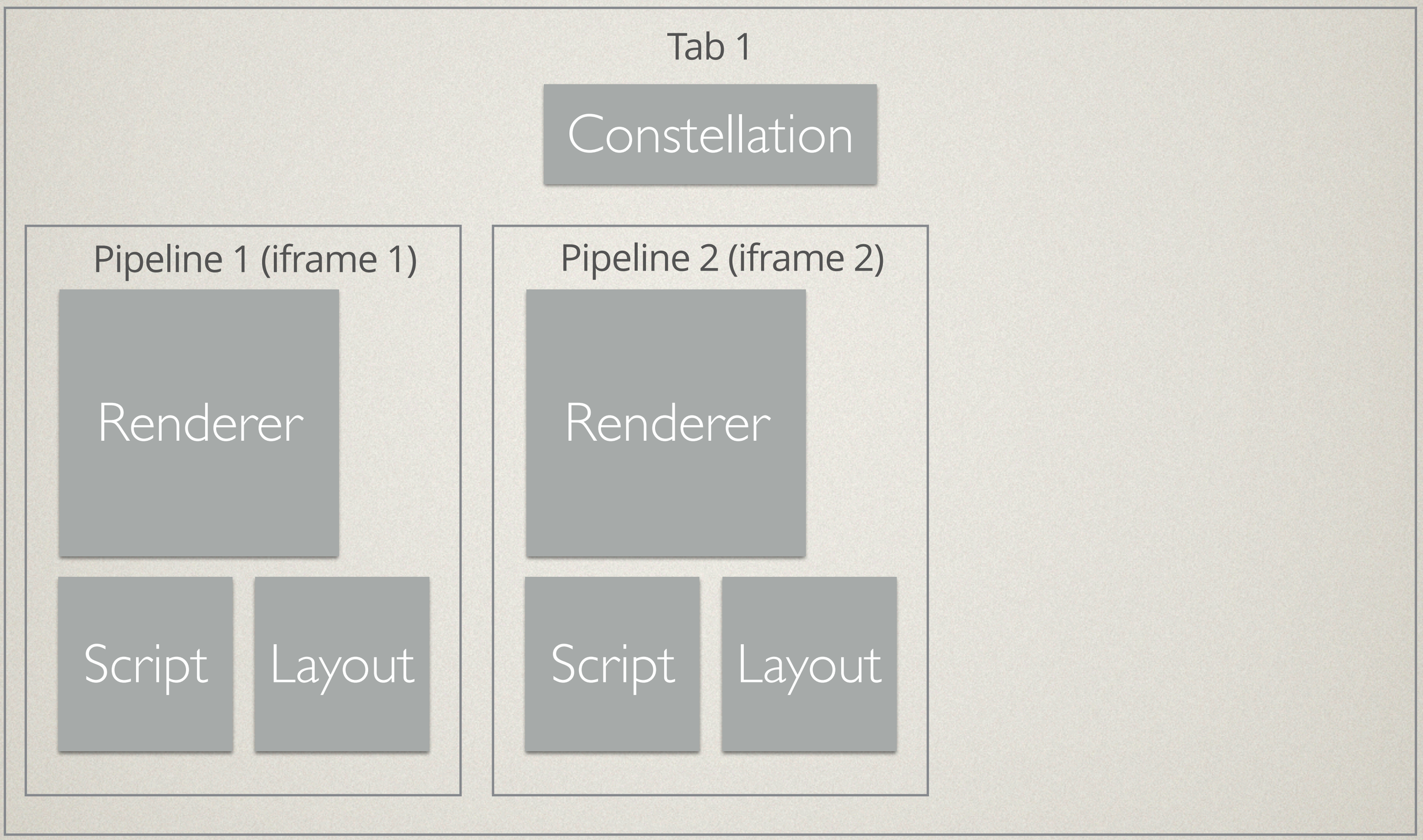




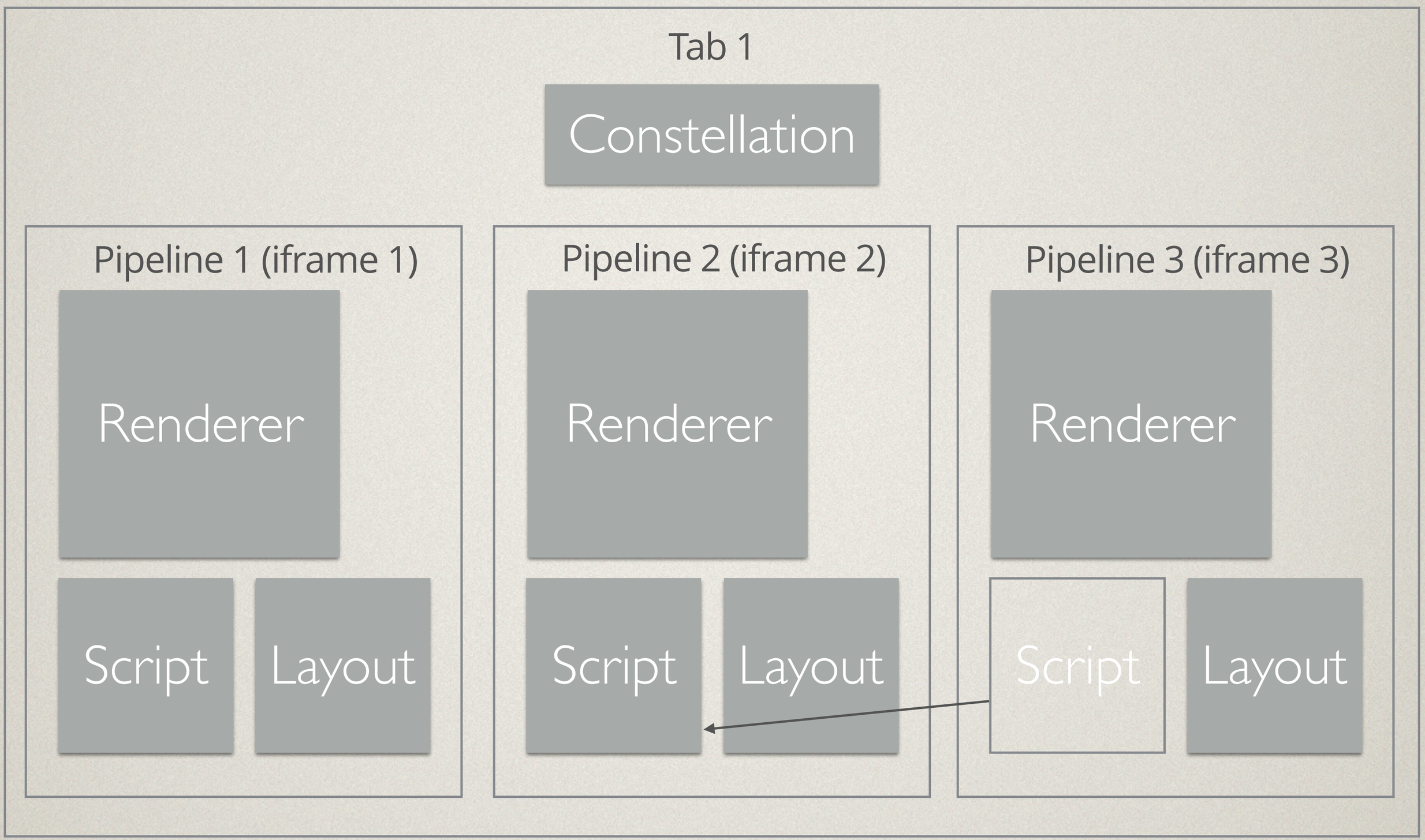
Servo's architecture



Servo's architecture



Servo's architecture



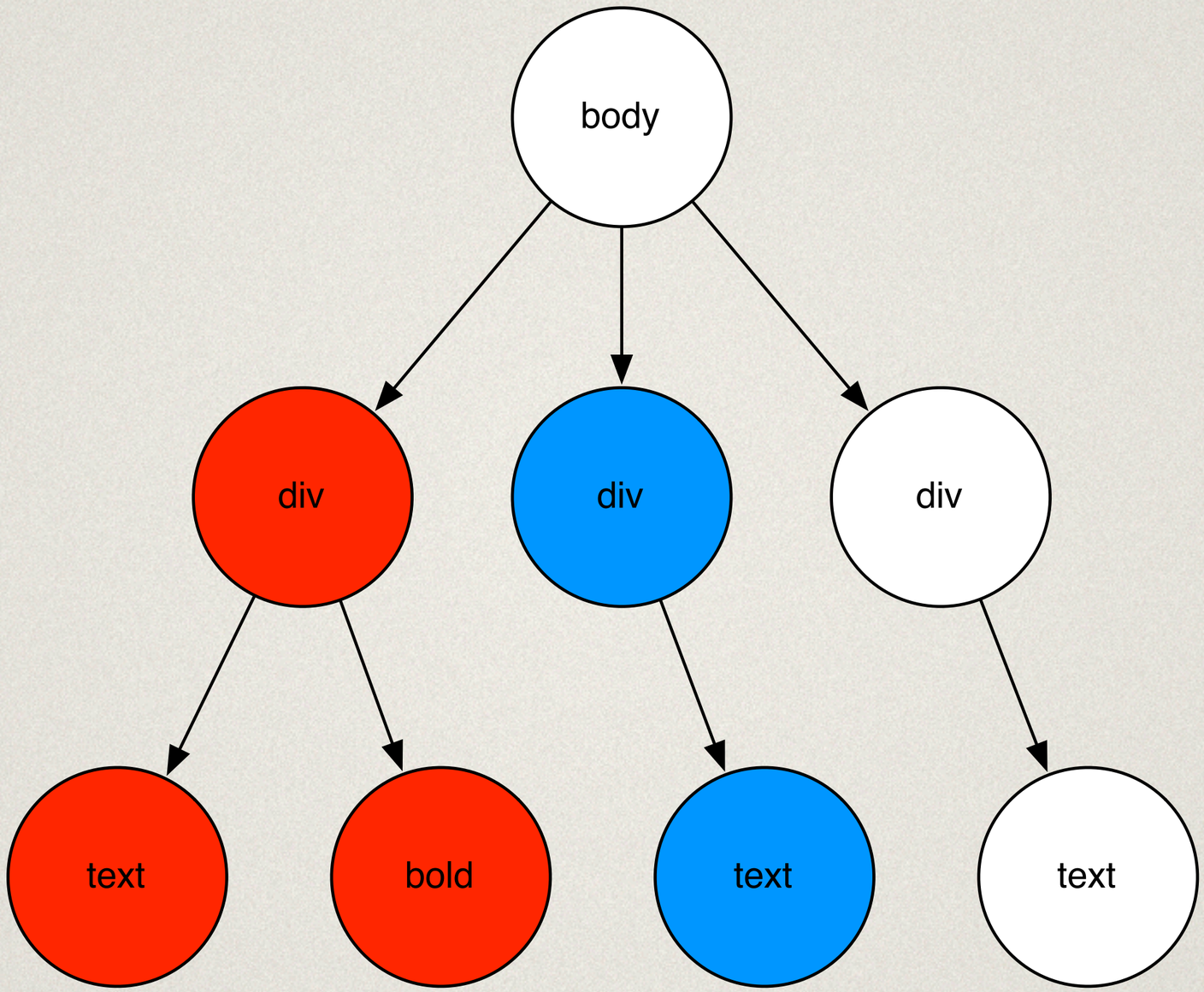
Demo: parallelism and sandboxing

Parallel layout

- Matters hugely on mobile platforms
  - Processors run at lower frequencies, but many cores
- Would enable more complicated pages on all platforms
- Implemented by work-stealing algorithm

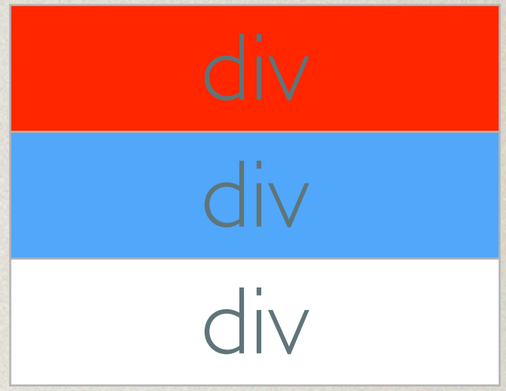
See: Fast and Parallel Webpage Layout. Meyerovich and Bodik. WWW 2010.

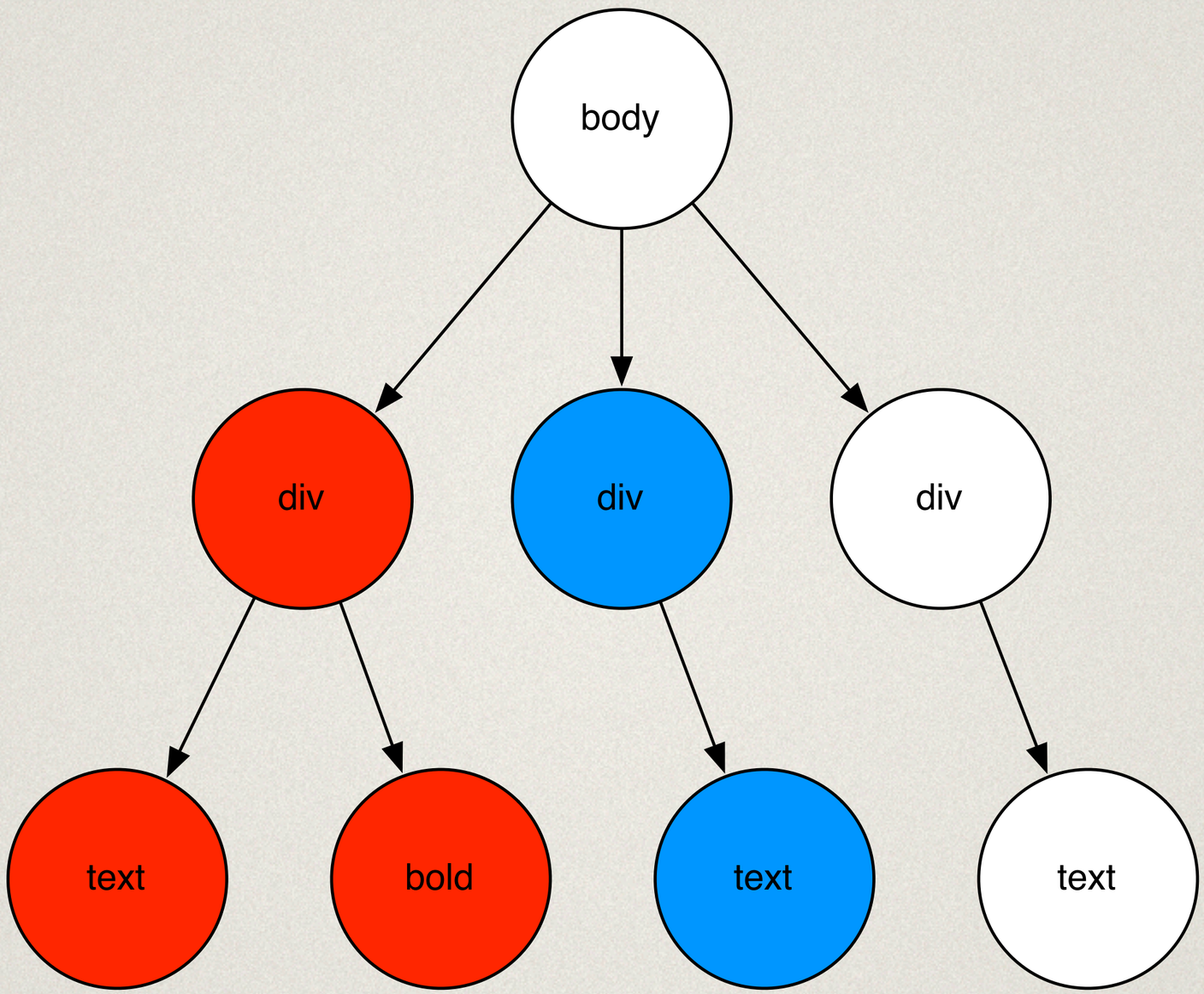
Parallel layout



Parallel layout

Queue

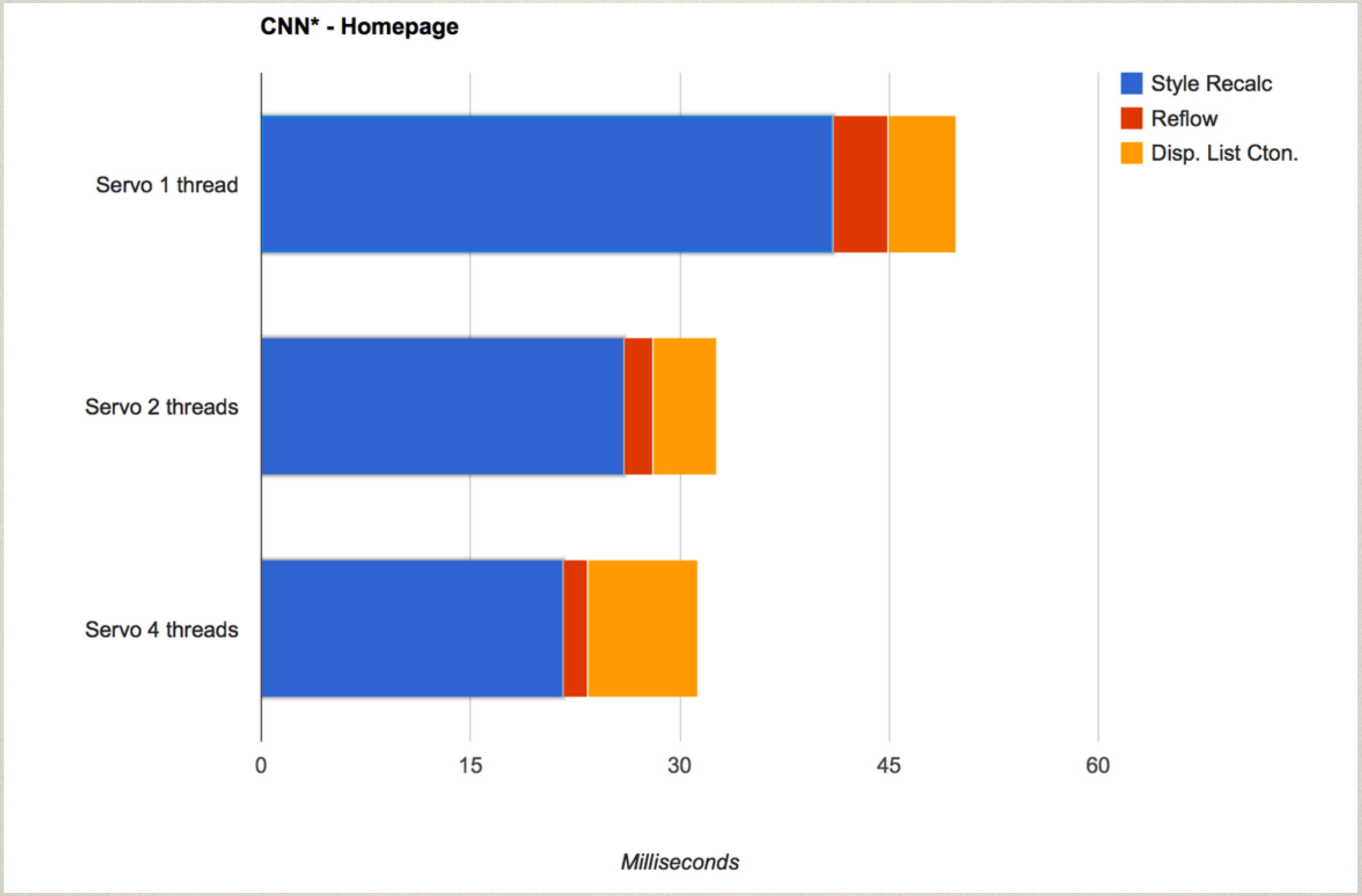




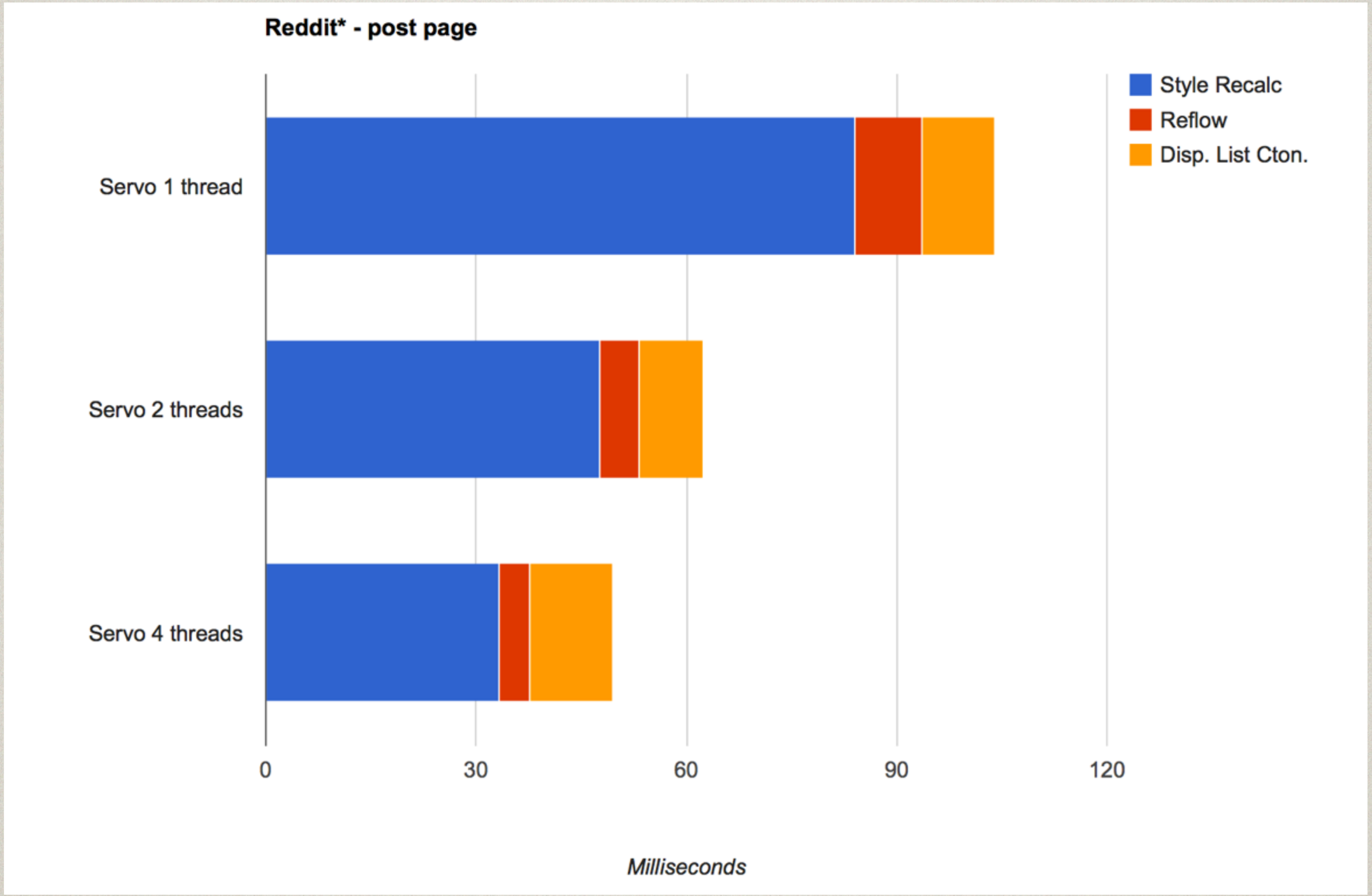
Parallel layout challenges

- HTML layout has complex dependencies
  - Inline element positioning
  - Floating elements
  - Vertical text
  - Pagination
- Considering adding speculation

Parallel layout speedups (CNN)



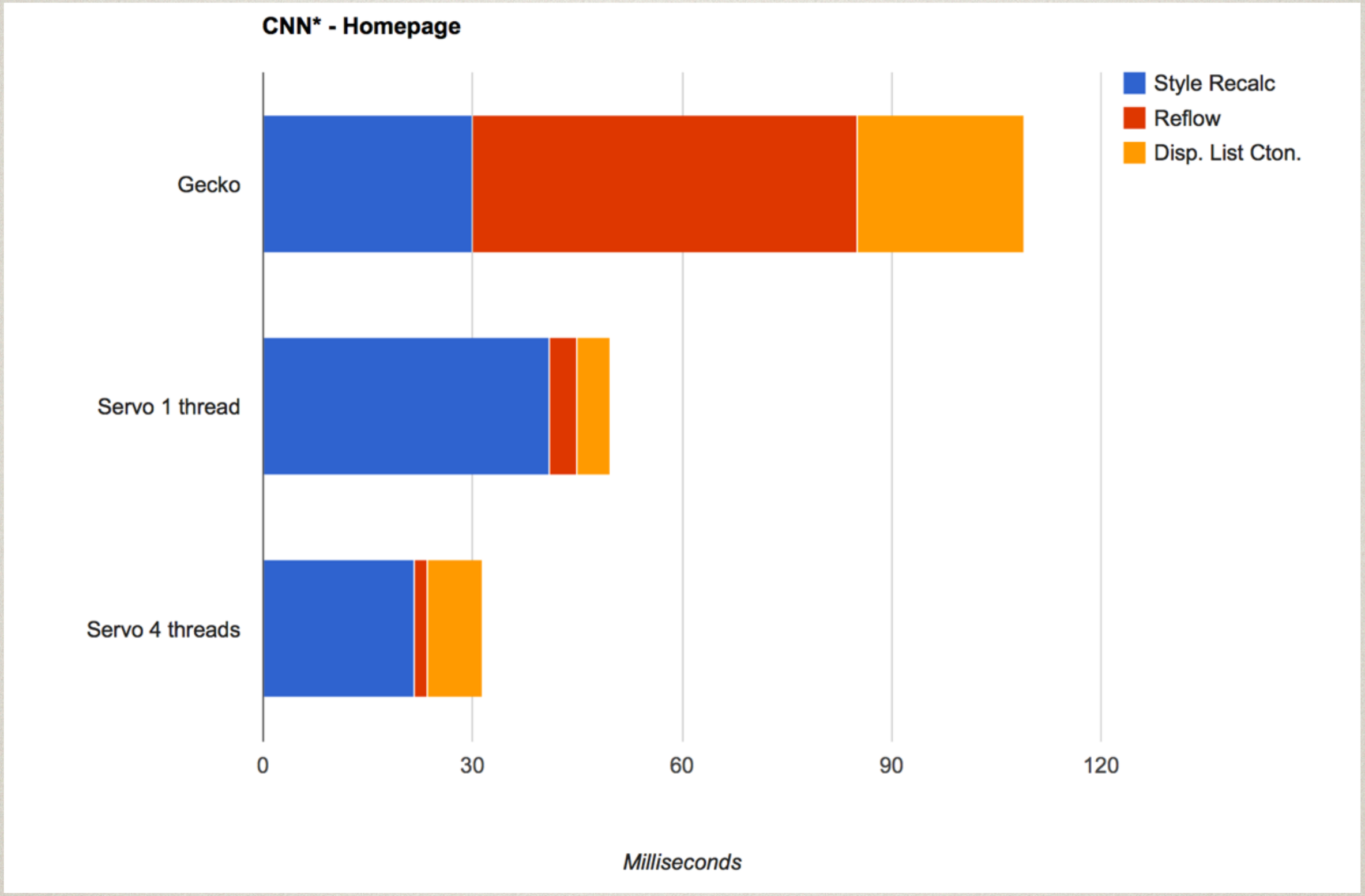
Parallel layout speedups (Reddit)



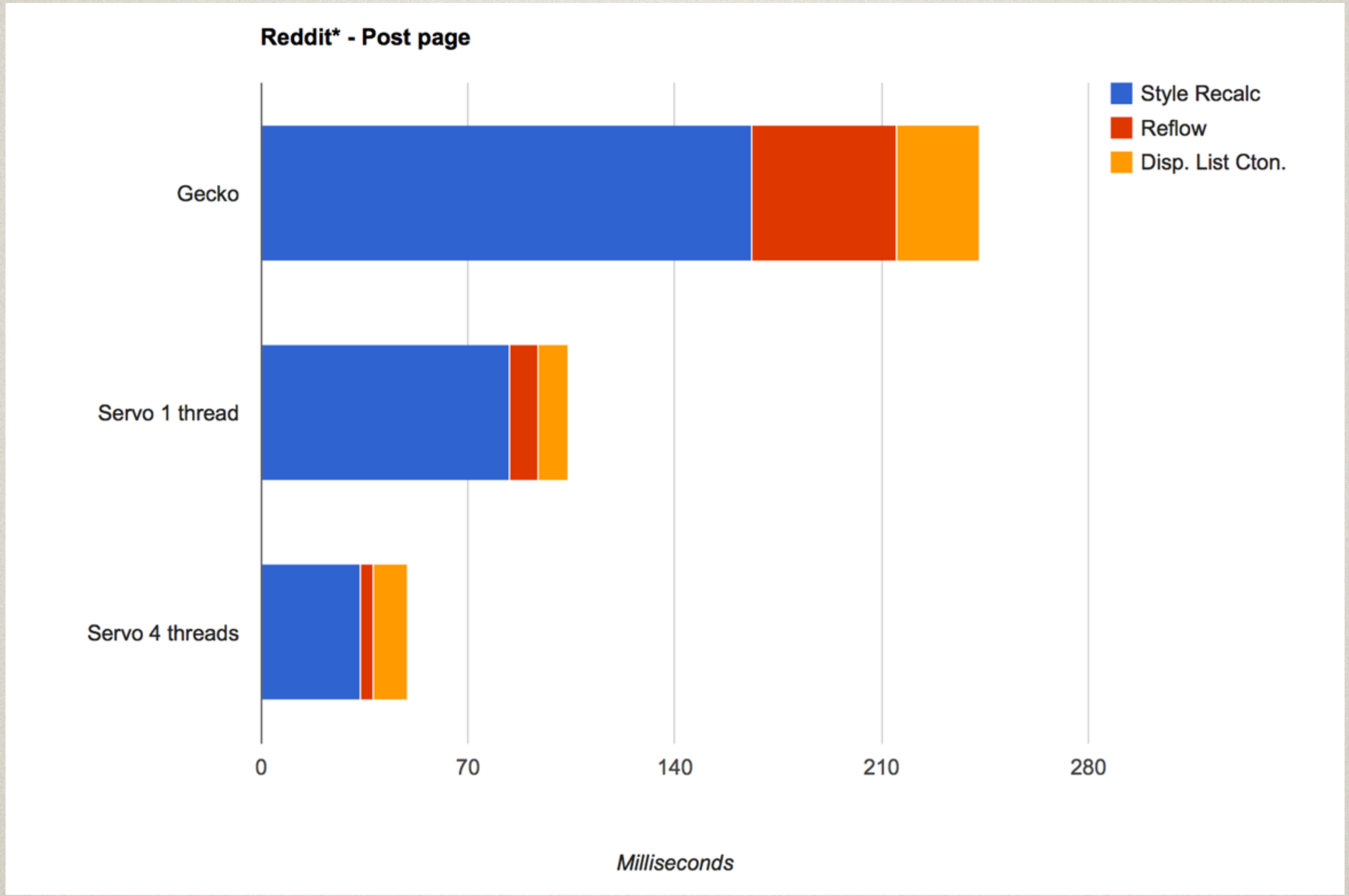
Aside: parallelism for power, too

- Force low-frequency CPU setting
  - Above four cores, same end-to-end performance as single core at high-frequency
    - BUT, 40% of the power usage
- Could also parallelize more
  - Rendering, CSS selector matching, etc.

Servo vs. Gecko (CNN)



Servo vs. Gecko (reddit)



From engine to browser

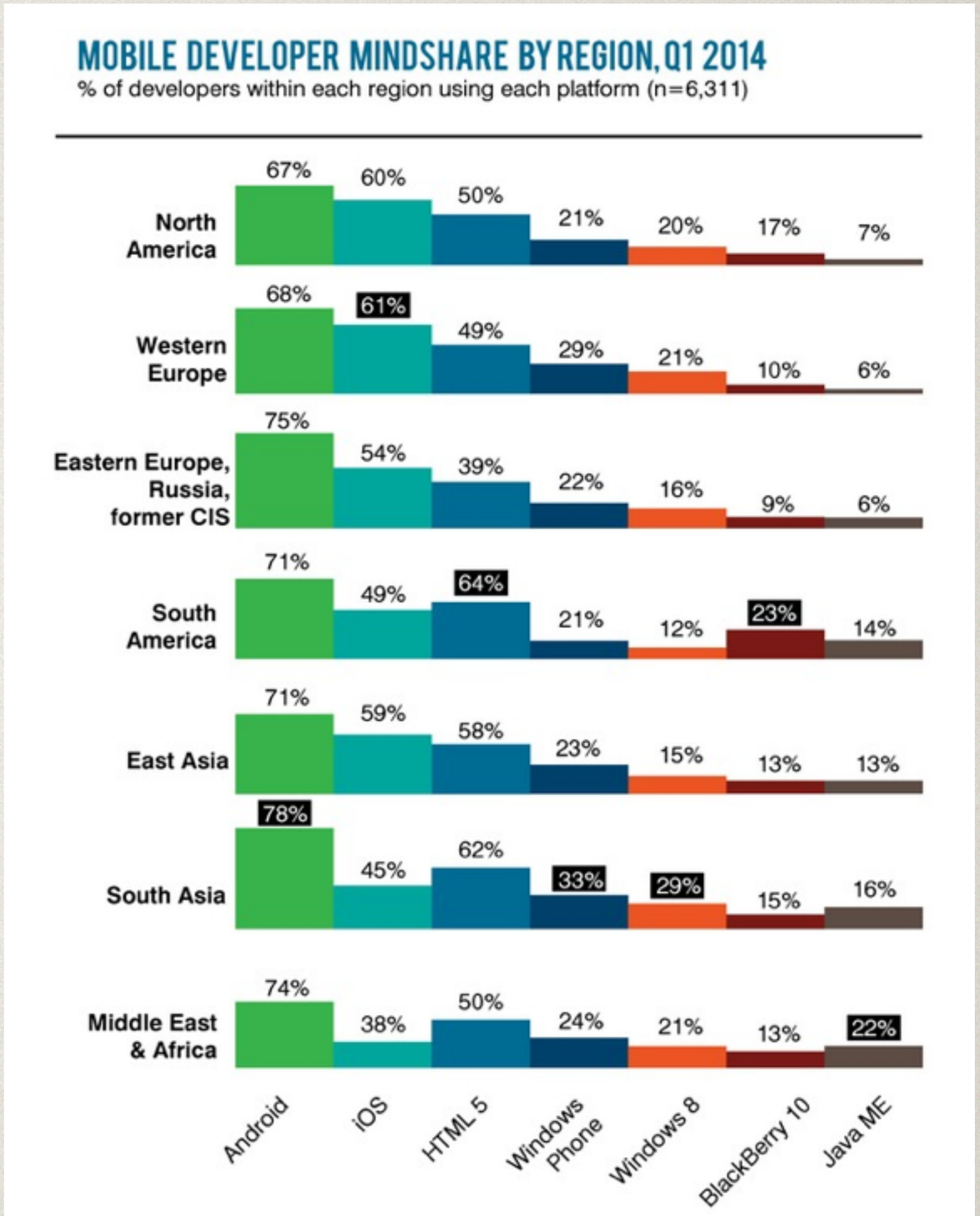
- Servo just renders pages
  - Similar to the Blink and Gecko engines
- Designed to work in many browser shells
  - Firefox OS, over interprocess communication (IPC)
  - Android, by implementing a Java wrapper
  - On the desktop with…

What is embedding?

- Hosting web engine in native application

Why embed?

- Reduced development time
- HTML5 popularity



% Highest regional Mindshare for the platform

Source: Developer Economics Q1 2014 | www.DeveloperEconomics.com/go

How not to embed

- WebKit
- Blink
  - Both suffer from an unstable API
- Application developer choices:
  - Ship full browser engine with application
  - Continually update to match breakages

How to embed?

- CEF: Chromium Embedded Framework
  - Isolates application developers from core API
  - C API with C++ extensions

Servo embedding strategy

- Stable API/ABI
  - Extensive API testing is a plus
- C-based
- Flexible
- Already designed

How to embed with Servo?

- Use CEF API+ABI
  - Removes need for YA embedding API
    - Less competition, more coding
  - Allows easy testing between engines
  - Servo: the pragmatic embedding engine

Servo embedding methodology

- Full symbol/ABI coverage
  - Every CEF function call resolves to a Servo function
  - Struct allocation sizes are identical

```c
typedef struct _cef_string_utf8_t {
    char* str;
    size_t length;
    void (*dtor)(char* str);
} cef_string_utf8_t;
```

C

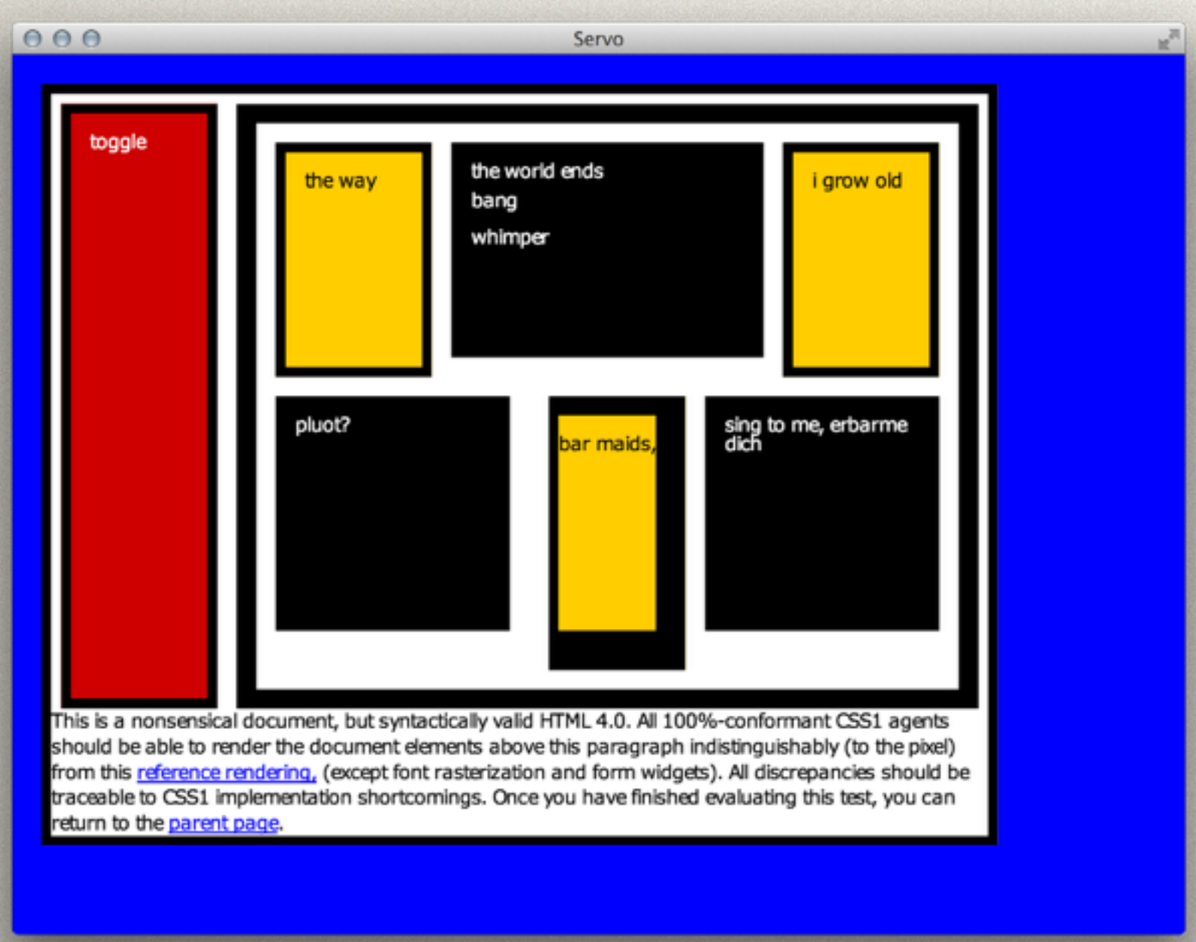
```rust
pub struct cef_string_utf8 {
    pub str: *mut u8,
    pub length: size_t,
    pub dtor: extern “C” fn(str: *mut u8),
}
```

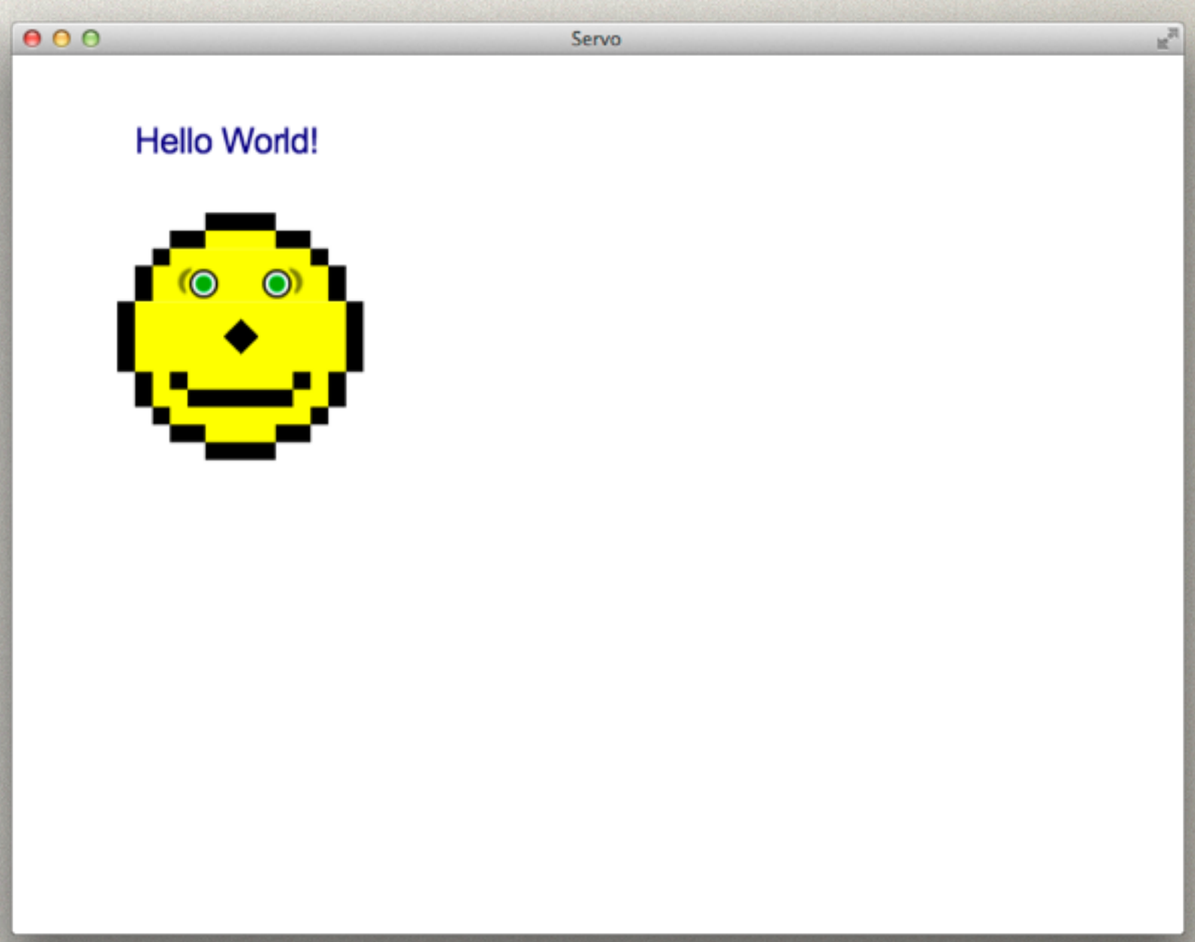
Rust

Servo embedding development

- Start with base set of symbols
  - `nm -u` on CEF applications
- Track function execution
  - CEF <-> Blink <-> Application <-> CEF ...
- Mimic CEF behavior using Servo equivalents
- Use preload hacks to test
  - LD_PRELOAD on Linux

Servo passes basic browser tests





Servo browsing demo!

Servo roadmap

- https://github.com/servo/servo/wiki/Roadmap
- Q4 2014
  - Improve dogfooding via CEF Desktop and native Android
  - Forms
  - Pagination
  - Graphics pipeline
- 2015
  - Try embedding Servo in Firefox Android & FFOS

Getting involved with Servo

- www.github.com/servo/servo/issues
  - Filter for “E-Easy”
- irc.mozilla.org, #servo channel
  - Worldwide community
  - Looking for more partners and contributors
- Hiring!
  - larsberg@mozilla.com
  - zmike@samsung.com