High-Performance Angular

Creating smooth and responsive applications in Angular, using case studies from MCP.

What do we mean by “performance”?

Fast Action Fulfillment

- How long does it take to fulfill user actions, especially ones that requires server data.

Smooth / Responsive

- How fast to we show feedback to user events/actions.
- Are transitions/animations smooth?

Action Fulfillment

- **Symptoms of slow action fulfillment:**
  - Waiting too long for some data or confirmation.
- **Examples:**
  - How long does it take to log in after submitting info?
  - How long does it take to load a loan/page after selecting one?

Responsiveness

- **Symptoms of poor responsiveness:**
  - Feedback for user actions is slow
  - Animations/transitions are not smooth/continuous
- **Examples:**
  - How long does megamenu take to show up after clicking button?
  - How smoothly does the page scroll?
  - Do spinners run smoothly or stop/skip?
  - Do hover/click reactions happen instantly?

Factor: CPU Thread

- Single thread (not counting worker threads)
- includes Painting / event handling

CPU Thread Profiling

- Performance profiler basic example

Factor: Network

- Browsers have a simultaneous connection limit. Any further requests made are queued until one completes.
  - Chrome: max 6 per hostname, 10 total
- Each network request needs to be handled individually by the browser, one-at-a-time, upon completion. If the CPU thread is doing something else then completion is queued.

Network Profiling

- Network tab and performance profiling example

Mitigation: Responsiveness

- React to user events quick enough.
- JS execution should be all in short bits, to allow for high FPS, at least whenever there are animations or transitions happening.
  - If we need to do a lot of CPU work, then break it up or put it in a worker thread. (usually we don't)
- Prioritization of user action feedback: React Fiber does this now

Mitigation: Action Fulfillment

- **Factors at play:**
  - Browsers have a simultaneous connection limit. Any further requests made are queued until one completes.
    - Chrome: max 6 per hostname, 10 total
  - API endpoints' turnaround time [server-side network bottleneck, outside scope]
  - Each network request needs to be handled individually by the browser, one-at-a-time, upon completion. If the CPU thread is doing something else then completion is queued.
    - Sometimes handling each request individually is more work than handling all at once, but almost never the other way around.

- **General Strategy:**
  - Minimize number of requests, get as much as possible done in each request.
  - Have enough CPU cycles available to react to initial event, and request completion

The CPU bottleneck: Change Detection

- React and Angular both have this bottleneck, since most of the work of a client side application is computing the next view state. This is a bit more complex in Angular because of 2-way data binding. React's simpler rules involving setState/props/shouldcomponentupdate make it a bit more intuitive
- This affects responsiveness directly, since we must do CD in order to update in reaction to user action.

How CD works in Angular: NgZone

- NgZone

How CD works: Tree Traversal

- OnPush
- This.changeDetectorRef.markForCheck()

CD performance strategy

- Make shorter and less frequent
  - Less frequent
    - Listen to less user events
    - Less network events
    - Less timeouts / intervals
    - Run things outside of angular
  - Shorter
    - OnPush
    - “render” code should be light (no javascript layout fn calls)

Examples

- With performance comparisons