**The 24th Twin Cities Code Camp**

Date: 4/15/2023

Address: Normandale Community College 9700 France Ave. S. Bloomington MN 55431

Attendants: 160+

1. Web App Observability
   1. Performance monitoring (Core web vitals)
   2. Synthetic Testing vs Real User Monitoring (Signal and noise in measurement)
   3. API monitoring – ensuring consistent outcomes
   4. Disease Control
      1. Website integrity
      2. Protection from systemic issues
      3. CSP – Content Security Policy
      4. Supply chain
   5. Interatrial Healthcare
      1. Client-side Observability, purpose: better UX
      2. Get the data, Level 0
      3. Share the data, Level 1
      4. Correlate the data Level 2
      5. Predict the data Level Next
2. No Silver Bullet: Use the Right Architecture
   1. Rockford Lhotka, Xspirit, <https://lhotka.net>
   2. Object Orientation, SOA and Microservices, Functional Programming (solve parallel and distributed computing)
   3. History: Monolith with VT terminal – Monolith on PC – 2 tier client/server monolith on PC – 2 tier client/server layered on PC – 3 tier client/server layered on PC – n-tier client/server layered on PC + cloud – SOA – Microservice (n-tier C/S on PC cloud) – Microservice (message-based async system of services)
   4. Layers Logical Architecture, Tiers or Services Physical Architecture, Cross-Cutting Concerns
   5. Separation of Concerns: Interface layer, Interface Control layer, Business layer, Data Access layer, Data Storage, Orthogonal Concerns
   6. Consumer <-> Interface, Interface Control, Business <-> Business, Data Access <-> Data Storage
   7. App Boundaries:
      1. Inside an app
         1. Organized functionality (layers)
         2. Deployment (tiers)
         3. Cross-tier communication is app choice
         4. Data ownership
         5. Trust across layers and tiers
      2. Between apps
         1. Apps are a “black box”
         2. Interact only with an app’s interface
         3. Cross-app communication is enterprise choice
         4. No cross-app data sharing
         5. No trust or coupling between apps
   8. Service-based System
   9. Best of Both Worlds
3. INP – Interaction to Next Paint
   1. Jeremy Wagner – jlwagner.net - @malchata
   2. Good < 200 ms, Needs improvement < 500 ms, Poor > 500 ms
   3. React – Preact, yield user input, prioritize work 0-9
   4. PostTask, Schedule.Yield
   5. Blocking tasks – input delay – Processing time (pointerup, mouseup, click) – Presentation delay (Render, Paint, Frame presented)
   6. Github.com/GoogleChrome/web-vitals
   7. Web.dev/optimize-long-tasks
   8. Web.dev/optimize-inp
   9. Optimize long tasks. Don’t block the main thread. Break up your long tasks.
   10. validateForm, showSpinner, saveToDatabase, updateUI, sendAnalytics
   11. Tasks scheduled with post tasks for the above 5 tasks
4. Redis
   1. @guyroyse – github.com/guyroyse/memory-first – guy.dev
   2. Memory first, NoSQL, Database (Persistable, Clusterable, Replicatable)
   3. A Giant Hash Table, Key to String, Bitmaps, Bitfields, Hashes, List, Sets, Sorted Sets, Geosets, Hyperloglog, Streams
   4. Redis is extensible: Modules, New data structure, New commands
   5. Redis stack (JSON, Search, Bloom, Graph, Time Series, Redis Insight)
   6. Talking to Redis
      1. PING -> PONG
      2. TIME -> “1679598546”, “805638”
      3. FLUSHALL -> OK
   7. RESP: The wire protocol
   8. Keys command is an evil. Don’t use it.
   9. Commands
      1. Nc -cv localhost 6379
      2. Redis-cli ping
      3. Redis-cli
      4. Redis.io/commands
      5. Redisinsight
   10. Redis
       1. <https://redis.io>
       2. <https://redis.com/redisinsight>
       3. <https://discord.gg/redis>