User Experience (UX)

Pankti Shah

Lean UX and Design Thinking

- Requirements Driven Development over Design Thinking

- Validate requirements with test cases before implementation

- Make decisions based on objective observations

- Extensive requirement gathering and validation process. Involves using tools like BALSAMIQ for prototypes

- This approach works for both new and existing products

High Quality Impactful UX Research

- Noticeability Test (User's actions over user's words)

- Impactful A/B Usability Test

- Rainbow Spreadsheets

Co-making Great Products

- Collaborative Agile development

- make sure the ideas work before making any investments

- don't create more code but impactful code

- Journey Maps

Lessons Learnt

- Case Study
- Collaborative Teams
- Recognizing assumptions to find a better fitting UI
- Invest in design process before implementation
- Focus on customer impact before development

Taming JavaScript

Tong Zou & Pankti Shah

Scalability and Resource Management

- Goal: Maintain/ Increase efficiency as the team grows
- Bootcamp: management, lifecycle, mentor (chain reaction), give smaller stories to each new hire
- Faster loading dev env; to make dev time more efficient
- Test driven development
- Automation testing

How can we use this?

- Bootcamp mentor
- Overall bootcamp process can be made more efficient
- Dedicated pair for each new hire?
- Keeping tests relevant up to date
- Refactor flaky and legacy tests
- Increase automation tests

Release Management

- weekly release cycle with daily content push
- dedicated release manager who was solely responsible for cherry-picks and code management
- run full test suite every time content push occurs
- bunch of technologies that they use post content pushes (Gatekeeper, test analyser, IRC bots (RevTracker), Perflab, HipHop, etc)
- developer needs to manually verify the changes that go out in content push
- branch deprecation every week

How can we use this?

- no down time releases would be one step closer to frequent release cycles
- we already have agile development cycle, 6 releases a year, with weekly content pushes

HTTP 2.0 / SPDY

-next revision of http, based on Google's SPDY protocol

-reduces web page load time by prioritizing requests and multiplexing responses

-no need for batching content (spriting,concatenating,sharding) to reduce RTT

-runs on TLS

-headers compressed by design

-reduces latency

How can we use it?

-It's baked into FF 11+ and Chrome, turn on apache mod_spdy

JS Performance Patterns

- JS Priorities: reducing load time (load async, etc), reduce DOM queries
- JS best practices: load JS async, batch DOM updates, keep functions small, scoped, monomorphic, clean up listeners, use js patterns like module, etc
- trust, but verify: always benchmark performance claims
- If necessary, then keep your shims/polyfills small and unobtrusive
- Testing: Chrome profiler to check for bottlenecks, yslow, jsperf, etc.
- use tracers and memory management tools, unit test

How can we use this?

- already following most of the best practices through code reviews
- we already use packaging, minification, gzipping, CDNs and selective loading to reduce RTT and # requests
- enforce and encourage more usage of Module Pattern in future

Questions ?