Week 1 Lecture 3

Business

Getting Ready

- http://www.joelonsoftware.com/items/2007/10/26.html
- http://alarm-clock.pseudoberries.com/

What's in this lecture?

- How to break down and schedule your day
- How to schedule building new functionality
- How to manage procrastination
- How to handle falling behind

Consistency

- Basic but need to be said again and again:
  - Start the day at the same time
  - Eat meals at the same time
  - Get outside in the sun
  - Workout/be active/be social
  - Stop work at the same time

Expectations for Day

- 10 % Research
- 10 % Communication
- 20 % Design and planning
- 30 % Coding and Testing
- 10 % Tweaks

Effective Research

- Documentation
- Join a technology's talk and core mailing list
- Check highest rated questions on StackOverflow
- Google-fu: <research-word> <technology> <method>
  good words: 'understanding' 'basic' 'example'
- Build the simplest example you can

Communication

- Balance between always being available and getting work done effectively
- Use the right technology for the right kind of communication:
  - Immediate: call
  - Participation: Skype/Chat/GoToMeeting
  - Other: Email

Email

- Three categories of relevant emails: archive | hold | follow-up
  - Archive: Out of mind/already handled
  - Hold: need more information before response
  - Follow-up: Needs response within the day

Email

- Other Tips:
  - Keep inbox empty!
    - Clear through ‘follow-up’ and ‘drafts’ every time you check your mail
    - Wait. People want useful replies, not knee jerk.

Chat

- Ask if interrupting
- Do not hesitate to ask if you can call
- Ask pointed questions

Design and Planning

- Clean sheet thinking: start with a blank page
- Write down ‘bite-size’ problem
- Write down three solutions (even if you think your first is awesome)
- Select best solution
- Write steps to execution (if not already done)
- Estimate Time

Coding and Testing

- Set aside a block of 4-6 hours where you do not have any interruption
- Try timeboxing: 50 minutes work, 10 break
- Test: learn to love print statements

Coding and Testing

- Cycle:
  Small Problem -> Research -> Planning ->
  Big Problem -> Break
- 2-4 hours long
- most important: remember to stop!

Tweaks

- Only once you have functionality complete
- Should be scratches, not surgeries

Inflation

- Realize that inflation can happen
- Do not seek complete understanding first time through
- Three cycles to reading:
  High-level view
  Basic Understanding
  Depth and expertise
- Intersperse with real code whenever possible!

Managing Procrastination

- Indicative of an underlying problem: have you been outside lately? Did you eat lunch?
- Write down the simplest functionality you can implement in 20 minutes.
- As soon as you get distracted or want to do something else, write it down
- This is a ‘distraction list’
- After 20 minutes, do everything on your distraction list you can do in 10 minutes

Falling Behind

- You're estimations are off
- Create a plan to get back on schedule
- Communicate to your team where you're at and what you're doing to get back
- No all nighters. Seriously. Working an extra hour for a week is much more effective