Code Spikes

When you gotta deep dive... bring a map

Exploring and testing the viability of solutions

Common Focuses

Think of tasks that you’re currently working on that would benefit from short term research and testing.

- Stack decisions
- Technology evaluation
- Code Refactoring
- API Integration
- Deployment Solution
- New code concept or paradigm to learn
- Optimization

Deliverables

If you don’t have a proof of concept, it didn’t happen.

- Prototype Implementation
- Performance Reports
- Integration Demo
- Data model
- Security assessment
- Algorithm comparison

List code spike task candidates | 15 min

Think of your current projects, what tasks need some further exploration before you start pushing code? **These should be specific to you and your project.**

- List 2-5 tasks that would benefit from being a code spike
- Note the importance of the tasks to the project:
    - **1** essential feature and/or is an essential dependencies
    - **2** essential feature but not an essential dependency
    - **3** not essential feature but is an essential dependency
    - **4** not an essential feature and also not an essential dependency
- What would be a useful deliverable?
- How long will it take you? (estimated)

10 minute break

Understand the problem

Break the problem up into smaller parts.

Discover the unknown unknowns

- Review official **docs**
- Use **search** terms
- Give yourself mental space to **explore**
- Pay attention to common **bugs**
- Don’t get into **details**

Plan the Code Spike

Knowing when to stop is half the battle

- Set your goals and objectives
- Define success
- Allocate time for:
  - Research
  - Experimentation
  - **Documentation**
- Have a review plan (with a colleague)

Research | Test | Document | Repeat

Plan a code spike | 30 min

- Pick a topic that intersects **priority** and **interest**
- Conduct **background research** and document:
  - Points of interest
  - Things you don’t know
  - Support sources
  - Potential problem areas
- Re-evaluate your objective to reflect your research
- **Break down what you need to learn into small elements (ordered if possible)**
- Re-evaluate your time estimation

15 minute break

Code Spike Cycle | 30 min

- Depending on your code spike outline, you may or may not get into much code.
- Plan to finish with some kind of deliverable
- Start with **research**
- **Test** something
- Write down the **results**
- If you feel off track but fixated, ask “so what” and evaluate if you’re on topic or not

Documentation

Write like someone’s gonna read it

- Use headings and point form - be concise and direct
- Make it easy to access
- Include links with clear descriptions
- Make sure there is a clear line between the objective and the summary

Share Results

The best research ever is meaningless if nobody knows about it

- Start with the problem you're trying to solve
- Highlight constraints
- Share the **"so what"**
- Avoid self-deprecation
- Be concise and back up words with proof of work

Present Results | 20 min

In groups of 3-4, take turns presenting your findings

- Share your **proof of work** and your **summary notes**
- Describe **the problem** that you are researching
- Direct your partners to look at **items of interest**

- **Ask questions** of one another (everyone should ask at least one question)
- **Give each other feedback** on the accessibility of your summary

Never forget the “**so what**” when sharing results

Summary & Debrief

Share your takeaways from today’s activities