Built it, but nobody came: Avoiding over-engineering

Jon Peck @fluxsauce | #badcamp 2016.10.22

Jon Peck

Architect, Four Kitchens

twitter.com/fluxsauce
github.com/fluxsauce
linkedin.com/in/jonpeck

What’s over-engineering?

... the designing of a product to be **more robust or complicated than is necessary** for its application, either to ensure sufficient factor of safety, sufficient functionality, or because of design errors. (Wikipedia)

Making something that isn’t used

Cautionary Tale

“Creative Works”

Old system had a librarian who managed taxonomy of movies, albums.

The librarian left, so editors reverted to flat tags.

PO for new site wanted order.

Migrated content into clean hierarchy, but...

PO left and the replacement didn’t share vision. Site launched.

Today? Flat tags.

What causes over-engineering?

Human nature - we want to do good things

Pride - because we can

Ignorance - don’t know of a better way

Don’t care - somebody else’s problem

Cautionary Tale

“Let’s reinvent Panels!”

Technical stakeholder at client concerned about performance.

Disallowed Panels, but...

Required Panels flexibility.

Reinvented the wheel, added dozens of hours, fragile & scary codebase.

When does over-engineering start?

Design - wireframes and mockups make it look easy

Meetings - priority miscommunication

Architecture - prescriptive requirements

Implementation - ignorance and inexperience

Success Story

“Event RSVP”

Site on a tight budget has an events calendar.

Wireframe has an RSVP.

Framework doesn’t have an off-the-shelf events calendar.

Where will RSVPs be handled?

Off-site; just link to it.

Analyze the request.

What is the **goal**?

Who **needs** this feature?

Why is this feature needed?

Is this the **only** way?

What **data** backs it up?

Success Story

“Minimum version of Internet Explorer”

Engineer assumed support of very old versions of IE.

Support means extra overhead.

Checked Google Analytics...

Less than 3%! HTML5 Shiv unnecessary.

Build a Minimum Viable Product

The **minimum viable product** (MVP) is a product which has **just enough features** to gather validated learning about the product and its continued development. (Wikipedia)

A MVP is the basis for iteration and drives the conversation.

Estimation, prioritization, iteration

Define the **cost** of a feature.

Cost and interest drive **prioritization.**

Iterate beyond the MVP.

Cautionary Tale

Editorial control over list items

Wanted *optional* arbitrary placement of items in a list of content.

Built it and the interface; worked on it until launch.

Really only needed a sticky flag.

Under-engineering

Iteration communicates by making something **tangible**.

If it can’t be used, it’s **not** iteration.

Iteration only works **if** there’s something to iterate on.

Cautionary Tale

“Site’s not built yet.”

Working on project for many months.

Most functionality was complete; content migrated, design was being implemented.

“Site’s not built, we can’t review!”

Navigation wasn’t finished; removed placeholders and finalized menu.

Did I really need to do that?

Spending hours making changes that have no tangible effect on the system.

Change for the sake of change.

Artificial challenges whose solutions don’t impact the product.

Slippery Slope

Version updates

Always apply security updates.

Always apply relevant bug fixes.

Incrementally update when convenient and testable.

Evaluate new features, upgrade if you need them.

Cautionary Tale

ESLint JavaScript errors

Site optimization.

ESLint produced list of all problems, including errors, warnings, and stylistic problems.

What should have gotten fixed — syntax errors and warnings.

What happened instead — entire rewrite of all JavaScript.

Slippery Slope

Micro optimizations

`for` vs `foreach`, `echo` vs `print`, `.` vs `,` …

No practical difference in most use cases.

Usually makes code harder to maintain.

Focus on big picture first — slow queries, caching, errors...

Do you have a story to tell?

Audience participation.

Avoiding over-engineering is not an excuse to say no.

Thank you.

linkedin.com/in/jonpeck
twitter.com/fourkitchens