# UP TO DATE

## MAINTAINING AND IMPROVING SHOPIFY THEMES

## Long live your code.

Be proactive instead of reactive.

# Past, present and future.

Meet your codebase ghosts…

## THE PAST

## Technical debt is unavoidable.

Does your team pay off their credit card every month?

## It’s all around us...

- Using third-party libraries that are now defunct
- Custom undocumented code added to a theme by an expert
- Themes that haven’t been updated in years
- Inherited code from other agencies/developers
- A “temporary” fix that was added months ago

## Legacy code is everywhere.

How much are you living with?

## Reaching the tipping point.

- Developers are scared to change anything
- Worried about the “Butterfly Effect”
- No one want to work on that store anymore
- Theme can’t be updated - bugs emerge
- App conflicts (with other apps and themes)

## I see dead parentheses.

Identifying your issues.

## Take stock of what you’ve got.

- Is it currently broken? Like really broken?
- When is the last time the code was touched?
- Does **anyone** understand what the code does?
- Are you using the latest Shopify features?
- What kind of improvements can be made?

## Be kind.

Attitude is everything.

## Focus on solutions, not blame.

- You don’t know what stresses previous developers were under/ time commitments, etc
- Software is built by humans and humans are NOT infallible
- Flip side is to not put code on a pedestal

## Why does it happen?

Identifying the path to technical debt.

| | Reckless | Prudent |
|---|---|---|
| **Deliberate** | No one has time to review it, so we'll just commit to master. | We know this bug exists, but we have to focus on this feature instead. |
| **Inadvertent** | What are JavaScript design patterns? | In hindsight, it's clear that we should have taken this approach instead. |

– Martin Fowler

## THE PRESENT

## Getting buy-in.

Is it worth the trouble to update or improve?

## What’s the return on investment here?

- Merchants may not recognize the importance of maintenance
- Avoid enabling the “Set it and forget it” attitude
- Sell them on the flexibility of new features and performance improvements
- Stress the importance of keeping things up to date (browsers update all the time)

## Help your team.

Reduce technical debt where you can.

## Document the heck out of everything.

- Documentation needs to be just good enough
- Document all the things - project with compilers, development process, etc
- Use comments in your code - eg. using liquid to create arrays
- Not fun, but super helpful when someone new joins the team

## Tooling.

- Linters, compilers, task runners
- Be wary because these can come with their own technical debt, but the pros generally outweigh the cons
- Linters can be tough with liquid - separate files

## Code styleguides.

- If you can’t implement a linter, this is the next best thing in the short term
- Be picky in pull requests (but nice!)
- Have one for CSS/JavaScript/HTML/Liquid, etc

## Establish some standards.

- Introduce some repetition and consistency in your code so that it is easier to ramp up
- BEM, SMACSS, OOCSS
- Explore design systems (eg. Polaris or custom)

## Know what has changed.

- Always use some form of version control
- Make pull requests small and associate branch with issue number
- Take your time with reviews and set aside time/ energy
- Establish a solid git branching model (eg. git-flow)

## Keep your merchants in the loop.

- Changelogs are easier for team to review than commit messages - also a marketing opportunity
- Let your clients know when updates have been made - stress importance of updating
- Listen to suggestions for features/improvements

## Happy devs, happy life.

- Nothing is too precious to be reworked and improved - encourage time to refactor
- Provide opportunities to work on new features
- Get team to take personal ownership of code that they are committing
- Don’t get too fancy, just solid readable code

## Incremental changes.

Small wins over time.

## Things you can do tomorrow.

- Remove any old polyfills that you no longer need (check out caniuse.com for browser support)
- Use resource hints (preconnect, dns-prefetch) to perform DNS lookups in the background
- Minify your HTML using liquid
- Run axe on your site to check for accessibility issues and make a plan to update

## THE FUTURE

## Look ahead.

Think about what your merchants are requesting.

## Be kind to your future self.

- Be ready to support new features down the line
- Not all “features” are optional (eg. a11y)
- Browsers change rapidly and new APIs appear that you will want to use
- There is no “set it and forget it”

## Keeping things fresh.

What should you be aware of?

## Font picker.

(maybe not for custom agency sites)

## Dynamic Payment Buttons.

SUPER popular.

## Sections and blocks.

Just scratching the surface
of how powerful these can be.

## Accessibility.

Mandatory updates.

## Internationalization.

All around the world.

## Stay up to date.

Bringing it all together.

## No code left behind.

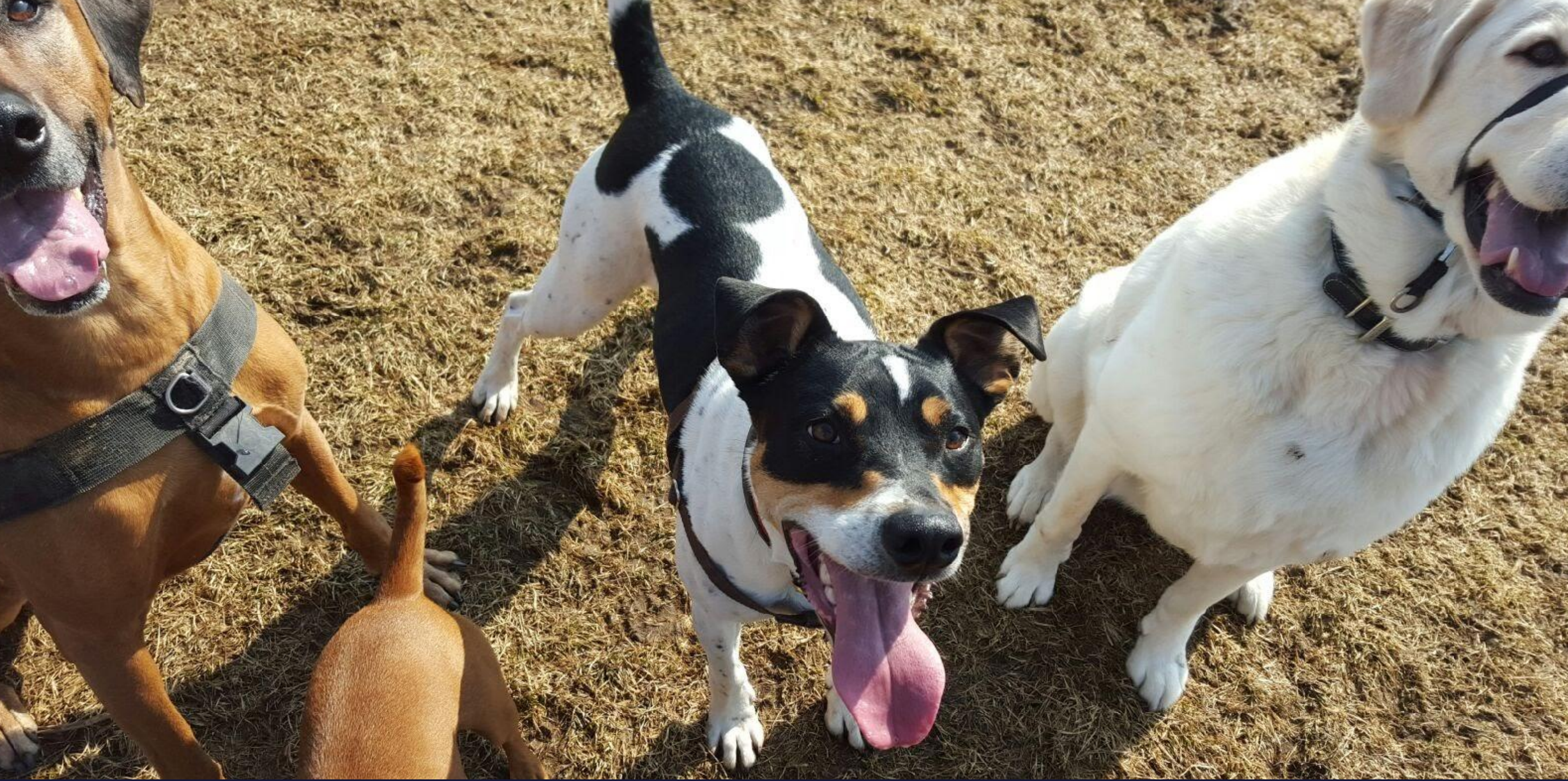
- If you are using a third-party theme, always aim to keep it up to date
- Sign up for mailing lists/ read development blogs
- Get involved with the Shopify community
- Attend conferences (good job!)
- Build time for maintenance into your schedule

## Maintenance is often overlooked

- It’s not hip, cool or sexy
- BUT being able to properly understand (and respect) legacy code is a skill
- It is critical and important

## Tenacity and optimism

- Don’t become complacent with the work you are doing
- Know your contribution affects the long-term prospects of the merchant and their business
- You’ve got this!



Thank you! @AlfalfaAnne @outofthesandbox