CHECKLIST:
BEST PRACTICES FOR CHOOSING OPEN SOURCE PACKAGES

Access to high quality open source packages helps developers bring new features, products, and services to market quickly. But finding the best packages is not always easy, and skimping on the research process can add maintenance headaches for your team down the road.

This checklist can help make your package research process smooth and effective.

- ☐ **Go with what you like.** If you've had a good experience in the past with a module, still vet it and make sure it's maintained, but as the saying goes, don't fix it if it isn't broken.
- ☐ **Don't limit your research to functional requirements.** Be sure to also consider things like the user experience impact modules can have. And make sure the package uses an open source license that is approved for use in your organization.
- ☐ **Determine if the package is well-maintained.** Answer questions such as:
  - ➜ Is there a regular cadence of addressing and clearing the issue backlog?
  - ➜ When was the latest release (not merge!)? Is there a release schedule?
  - ➜ How many maintainers work on the package? How many collaborators?
  - ➜ Do the maintainers work to a plan? This is a common attribute of strong projects.
- ☐ **Pay attention to how the maintainers interact with the community.** Chances are good that at some point you'll need a patch merged or will want help with a new feature. Are maintainers responsive and receptive to such requests?
- ☐ **Get comfortable with the package security posture.** Check for a vulnerability disclosure policy and secure DevOps best-practices. For examples of the kinds of things you'll want to be looking for, here is a list of the security tasks we ask maintainers to complete for packages that are included in the Tidelift Subscription.

For additional tips like these, download The Tidelift guide to choosing open source packages well.

CHECKLIST:
OPEN SOURCE LICENSING BEST PRACTICES

With modern software applications constructed using dozens to hundreds of open source package dependencies, it is critical to understand the licenses all of these projects use. Equally critical is maintaining development velocity, and achieving both of these aims simultaneously can be tricky.

This checklist can help you get a better handle on the open source licenses you use.

- ☐ **Get the lay of the open source license landscape.** In addition to the Tidelift guide to working with open source licenses, here are a few other good resources:
  - ➜ Succinct overview of the most popular open source licenses from GitHub at choosealicense.com
  - ➜ Comprehensive reviews:
    - The Legal Side of Open Source
    - Using Open Source Code, from The Linux Foundation
  - ➜ Want to go even deeper? Here's the definitive book on the subject: Open (Source) for Business: A Practical Guide to Open Source Software Licensing — Second Edition
- ☐ **Create a whitelist of approved open source licenses for your development team to reference.** The TODO Group has assembled a good list of example open source license policies.
- ☐ **Consider appointing a member of the development team as open source license czar.** This person takes on the role of staying current on the topic and answering questions about package license ambiguity from the rest of the team. They should follow IANAL (I am not a lawyer) best practices and bring in a member of the legal team when needed.
- ☐ **Talk as a team about why licenses matter.** Discuss why they matter and why everyone should check licenses as a standard part of their package research process.
  - ➜ This is a good primer presentation to use for a lunch and learn
- ☐ **Do a license audit.** As your code and customer bases grow, an audit can help you get a fuller view of open source packages you use and how they are licensed. Look for ways to do this without sacrificing developer productivity.
  - ➜ Tidelift offers comprehensive license scanning of your exact open source dependencies as part of our subscription. There are also standalone scanning tools, some free some for a fee, on the market and language-specific tools are also emerging, such as the npm license checker for JavaScript
- ☐ **Prioritize refactoring code with problem licenses in regular sprints and other backlog burndown.** Resist the temptation to kick the can down the road on code issues that don't directly impact users. Code with a problem license can become a ticking time bomb. Don't wait for the customer, or acquirer, code audit.