Epic Programming Principles Cheat Sheet

Kent C. Dodds www.epicweb.dev/principles

Core Philosophy

Core beliefs behind the approach to coding, relationships, and responsibility.

- Software is built for people, by people
- Value in-person connections
- Be the kind of person people like working with
- Do good in the world
- Value your time
- Don't solve problems, eliminate them

Debugging & Resilience

Principles for debugging, error handling, and building resilient systems.

- Least privilege
- Design to fail fast and early
- Optimize for the debugging experience
- Install extinguishers before the fire starts

Testing & Performance

Guidelines for testing, performance optimization, and quality assurance.

- Tests should resemble users
- Make assertions specific
- Explicit is better than implicit
- Weigh the cost-benefit of performance optimizations

Craft

Development standards, best practices, and decision-making processes.

- 'Best Practices' do not exist
- Do as little as possible
- Make it work, make it right, make it fast
- Optimize for sustainable velocity
- Optimize for the unknown
- Pragmatism is more important than 'purity'
- Keep it consistent
- Don't confuse simplicity with familiarity
- Balance innovation with stability
- Take ownership
- Use Static Testing Tools
- Don't prioritize temporary problems over long-term problems
- There is no such thing as the 'right tool for the job'
- Default to standards
- Prioritize tools with an ecosystem

Developer Experience

Principles for improving the development process and team productivity.

- Adapt to and adopt productive tools
- Document your work
- Enable Offline development
- Deployable commits
- Small and short lived merge requests
- Go down to level up

Career

Teamwork, ethics, communication, and career development.

- Solidify knowledge through teaching
- Magnify your conversation impact
- Communicate value
- Know where to make money
- Focus on unique value proposition
- Keep learning
- Prioritize relationships
- Strive for excellence
- Be honest
- Embrace reality
- Take personal responsibility

Architecting your project

- Avoid tight coupling to dependencies
- Favor composition over inheritance
- Avoid Hasty Abstractions (AHA)
- Separate concerns
- Don’t synchronize state, derive it
- Use the Principle of Least Surprise