**Programação de melhores práticas Tidbits**

Uma coleção de citações e parafrases para desenvolvedores de toda a web.  
Use seu próprio julgamento em sua aplicação.

**Foco nas estruturas de dados**  
Os programadores ruins se preocupam com o código. Os bons programadores se preocupam com as estruturas de dados e seus relacionamentos  
- Linus Torvalds  
Fonte

**Nunca crie grandes aplicativos**  
O segredo para construir grandes aplicativos nunca é construir grandes aplicativos. Quebre suas aplicações em pequenos pedaços. Em seguida, monte essas peças testáveis ​​e mordidas em sua grande aplicação.  
- Justin Meyer, autor JavaScript MVC  
Fonte

**Aceite que você não tem idéia de como isso vai crescer**  
A chave é reconhecer desde o início que você não tem idéia de como isso vai crescer. Quando você aceita que não sabe tudo, você começa a projetar o sistema defensivamente ... Você deve passar a maior parte do tempo pensando em interfaces e não em implementações.  
- Nicholas Zakas, autor "sites de JavaScript de alto desempenho"  
Fonte

Reconhecimentos para Addy Osmani

**Siga os princípios de X**  
Fonte: https://github.com/thomasdavis/best-practices  
Em 1984, Bob Scheifler e Jim Gettys estabeleceram os princípios iniciais do X:  
• Não adicione novas funcionalidades a menos que um implementador não possa completar um aplicativo real sem ele.  
• É tão importante decidir o que um sistema não é como decidir o que é. Não sirva todas as necessidades do mundo; Em vez disso, faça o sistema extensível para que as necessidades adicionais possam ser atendidas de forma compatível com o alto.  
• O pior do que a generalização de um exemplo é generalizar a partir de nenhum exemplo.  
• Se um problema não for completamente compreendido, provavelmente é melhor não fornecer nenhuma solução.  
• Se você conseguir 90% do efeito desejado para 10% do trabalho, use a solução mais simples. (Veja também Pior é melhor.)  
• Isolar a complexidade o máximo possível.  
• Fornecer mecanismos e não políticas. Em particular, coloque a política da interface do usuário nas mãos dos clientes.  
O primeiro princípio foi modificado durante o design do X11 para: "Não adicione novas funcionalidades a menos que você conheça algum aplicativo real que exigirá isso".  
Fonte  
**Questões de Qualidade**  
Quando eu ouço "APENAS BANG OUT CODE QUE TRABALHA", penso em todos os aplicativos que eu não uso mais porque gradualmente perderam a capacidade de iterar.  
- Avdi Grimm  
Fonte

**Não faça coisas difíceis, faça coisas fáceis.**  
• Simples é melhor do que complexo.  
• Complexo é melhor do que complicado.  
• Flat é melhor do que aninhado.  
• Contagens de legibilidade.  
• Se a implementação é difícil de explicar, é uma má idéia.  
• Se a implementação é fácil de explicar, pode ser uma boa idéia.  
- O Zen of Python  
Fonte

Shortlist cortado da palestra "Pare de Escrever Classes" de Jack Diederich

**Não escreva código**  
Não escreva código (escreva novo código apenas quando tudo o resto falhar) é a lição mais importante que cada desenvolvedor precisa aprender. A quantidade de código duplicado e crappy (entre projetos) que existe hoje é esmagadora. Em muitos casos, os desenvolvedores nem sequer se preocupam em olhar ao redor. Eles só querem escrever código.  
Fonte

**Reduzir a quantidade de código em seu produto deve ser um objetivo.**  
"Eu odeio o código, e eu quero o menor possível no nosso produto". - Jack Diederich  
Fonte  
Mantenha Dependências Lean  
O velho ditado "não reinventar a roda" não se aplica quando a roda vem ligada a um motor de locomotiva.  
Fonte

**Pare de escrever aulas**  
A assinatura de "isso não deve ser uma classe" é quando a classe tem dois métodos, e um deles é o construtor. Toda vez que você vê esses sinais, você provavelmente deveria ter escrito uma função.  
- Jack Diederich  
Fonte

**Reinventar a roda**  
Inventar suas próprias rodas dá-lhe uma profunda apreciação e compreensão de como as rodas funcionam e o que faz uma boa.  
Fonte

**Refatoração> Reescrevendo**  
Desculpas comuns para uma reescrita de software  
1. O código suga  
2. "Nós somos muito mais inteligentes agora"  
3. Escolhemos a Plataforma / Linguagem errada  
Por que reescrever é (quase) nunca uma boa idéia  
1. Toma sempre o tempo que você espera  
2. Mudança de Mercados  
3. Clientes existentes ficam frustrados  
4. Refatoração pode limpar o código  
5. Você não controla a reescrita, ele controla você  
Fonte

**Reescrever> Patching**  
Se você estiver mudando mais de 25% de uma classe ou método, considere simplesmente reescrevê-lo. Você escreverá o código de forma mais limpa.  
Esqueça novos recursos, basta fazer as mesmas coisas melhor.  
O problema: é muito fácil perder de vista o que os usuários muitas vezes se preocupam com mais, qual é o desempenho e a usabilidade das aplicações e recursos que eles já usam com mais freqüência.  
- Tim Anderson  
Fonte

**Escreva testes de unidade**.  
Todo programador sabe que eles devem escrever testes para seu código. Poucos fazem. A resposta universal a "Por que não?" É "estou com muita pressa". Isso rapidamente se torna um ciclo vicioso: quanto mais pressão você sentir, menos testes você escreve. Quanto menos testes você escreve, menos produtivo você é e menos estável seu código se torna. Quanto menos produtivo e exato estiver, mais pressão você sente.  
Os programadores queimam apenas desses ciclos. Separar exige uma influência externa. Encontramos a influência externa que precisávamos