我认为十件重要的事:

1. **理解业务目标高于技术实现**
   架构必须服务于业务需求。脱离业务目标的复杂技术方案会导致资源浪费和项目失败。例如，过度追求微服务可能增加维护成本，而单体架构可能更适合小型团队。
2. **技术债务管理**
   忽略技术债务会引发"破窗效应"。Netflix曾因未及时重构缓存系统导致全球服务中断。架构师需建立债务评估机制，平衡短期交付和长期维护。
3. **非功能性需求（NFRs）的优先级**
   性能、安全、可扩展性等NFRs常被忽视，但可能直接导致系统崩溃。2017年Equifax数据泄露事件正是安全NFR处理不当的典型案例。
4. **架构决策的可见性**
   关键决策需要文档化并通过架构决策记录（ADR）保存。Uber早期未记录服务拆分决策，导致新团队重复踩坑。
5. **模块化与解耦设计**
   高内聚低耦合的模块设计能提升系统弹性。Amazon通过将单体架构拆分为服务化架构，支撑了业务指数级增长。
6. **持续演进而非完美设计**
   Spotify采用"增量架构"策略，通过每周架构审查会议实现系统渐进式改进，避免了大规模重构的阵痛。
7. **技术选型的长期成本**
   盲目追求新技术可能带来隐藏成本。某金融公司采用GraphQL时未考虑团队技能储备，最终实施成本超预算300%。
8. **风险管理框架**
   架构师应建立风险登记册，量化评估风险。NASA的火星气候探测者号事故（单位制不统一导致坠毁）印证了风险控制的重要性。
9. **数据设计先行原则**
   数据模型决定系统扩展边界。Twitter早期未设计好推文分片策略，导致2010年大规模服务中断。
10. **架构可观测性设计**
    预先设计监控埋点比事后补救更高效。Google的Dapper系统通过在架构层面集成追踪，将故障定位时间缩短了80%。