# 前端架构八大设计准则

1. **适度设计（第一准则）**
2. 架构设计应以满足一定周期内的需求为目标，周期一般考虑一年即可，需求包括功能性与非功能性（优化，扩展等）。在这两方面都满足，并考虑一定的前瞻性的前提下，架构应尽量简单：降低成本，缩短实现周期，使质量更高。

i 能少一个组件就少一个组件

每增加一个组件，包括研发、测试、运维、硬件资源占用的总体成本就可能会上升数万元

ii 能不用微服务就不用微服务

微服务是有一些有点，但缺点也很明显：响应时间增加、综合成本上升、管理复杂度高。对于中小型系统，不需要频繁发布的系统，采用微服务架构就像是杀鸡用牛刀。

1. **避免问题扩大化**
2. 解决一个问题时尽量直接解决，不要间接解决而引入新的问题

i. 举个例子，当我们用A插件的时候，发现它存在一些问题，然后我们又引入了B插件来解决这个A插件存在的问题，以此类推，正确的做法是直接把A插件还掉，避免引发新的问题

1. **本质思考**
2. 作为架构师，应当有广博的知识面，能够透过现象看到问题的本质。

脑海中印象很深的一件事是，有一次我们做了一个文件的上传功能，刚开始有用户反馈上传速度太慢了有时候还上传失败，大家一致认为是用户的网络有问题，直到后面又有客户反馈这个问题，我立马就意识到，这肯定不是网络的问题，而是单文件过大，IIS的限制导致问题的发生。

后续的方案是采取了分片上传、修改IIS限制，解决了棘手这个问题。

1. **技术选型以需求为主，而不是人员现状**

A）经常看到这样的情况，在做技术选型决策是，明知某种技术是最优的，但还是选择当前团队掌握的技术，这种决策可以理解，但对产品长远发展不利，极有可能在竞品的PK中落于下风（性能以及客用户体验上）。

B）技术行业最经典的一句话：没有解决不了的技术问题，只有解决不来问题的人。

**始终坚信项目是可以推动团队技术发展的。**

1. **重视成本**
2. 很多架构师可能有这样一种思维，把架构设计得简单了，会显得自己很平庸，弄一个高大上的架构出来，方显本事，在领导面前述职时也有善可陈。可是，如果太超前去搞这种复杂的架构，会白白增加成本。
3. 架构师的每一个决定都会影响到成本。包括研发成本，第三方软件硬件采购成本和上线后的运营成本。不同的方案，在成本分布和最终的总体成本上会有不尽相同。往往需要综合权衡，才能决策出成本最优的方案。
4. 成本关系着企业利润，最终也会影响到个人奖金。这种因果关系的提现周期虽然较长，也不是很透明直接，但道理不会错。身为架构师，没做一个决策前，先问问自己，这个方案会花费多少成本。
5. **既要有拿来主义， 也不能生搬硬套**

A）有的人做设计，完全就是拿来主义，不从自己系统的实际情况出发。而另一种人，则完全自己发挥，不去调查类似系统的设计方案。这两种情况都不可取，应该从自己系统的实际需求出发，在参与别人成熟案例的基础上，少走弯路，并考虑自己系统的特点，给出符合自身需求的设计。

1. **重视流程（项目的一致性、可维护性）**
   1. 现在早就过了FTP上传文件的时代，那么现在重要的是思考怎么用工具和流程构建一个高效且避免出错的工作流。
   2. 工作流变得越来越复杂，那些用户它们的工具也同样如此，这些工具在提高生产力、加快效率和保持代码一致性上带来了惊人的效果。比如说环境的准备，有测试环境、生产环境、备用环境等等，各个板块是互不关联的，出错了也不用慌，有备用版本。
2. **性能测试与宏观建设**
3. 不满足于业务功能的实现，而是要考虑到性能与响应速度。
4. 网站性能基线与行业水平和通用的最佳实践相比较是必不可少的。
5. 推荐 HTTPArchive（http://httparchive.org/），它测试并记录了几十万个网站的性能数据。
6. 数据分析

i.页面大小：2061KB

ii.总请求数：99

Iii.可缓存资源所占比例

1. 如果想让自己的网站比大部分网站都快，可以考虑设定一个目标：一个1648KB的网站，包括79个请求，其中44个请求可以缓存，这将使你的网站领先平均水平20%