**Git提交信息规范化**

**目的**

* 统一团队Git Commit标准，便于后续代码review、版本发布
* 可以提供更多更有效的历史信息，方便快速预览以及快速合并代码；
* 团队其他成员进行类git blame时可以快速明白代码用意；

**Git基本使用**

关于代码clone、pull、commit、push、merge、create new branch等操作，本次暂不做描述，可以自行查阅。

**Git版本规范**

**分支**

* master分支为主分支(保护分支)，不能直接在master上进行修改代码和提交；
* develop分支为测试分支，所以开发完成需要提交测试的功能合并到该分支，最终发布代码用的也是此分支；
* feature分支为开发分支，大家根据不同需求创建独立的功能分支，开发完成后合并到develop分支；
* fix分支为bug修复分支，需要根据实际情况对已发布的版本进行漏洞修复；

**Tag**

采用三段式，v版本.里程碑.序号，如v1.2.1，按照当前app的发布版本进行设定。

**changelog**

版本正式发布后，需要生产changelog文档，便于后续问题追溯。目前我们没有这样做。

比如：[HandyJson的changelog](https://github.com/alibaba/HandyJSON/blob/master/CHANGELOG.md)

**Git提交信息**

message信息格式采用目前主流的Angular规范，这是目前使用最广的写法，比较合理和系统化

type(scope):subject

* type：用于说明commit的类别，规定为如下几种
  + feat：新增功能；
  + fix：修复bug；
  + docs：修改文档；
  + refactor：代码重构，未新增任何功能和修复任何bug；
  + build：改变构建流程，新增依赖库、工具等（比如工程文件的更改）；
  + style：仅仅修改了空格、缩进等，不改变代码逻辑；
  + perf：改善性能和体现的修改；
* scope：【可选】用于说明commit的影响范围
* subject：commit的简要说明，尽量简短

