atitit 编程语言之道attilax著.docx

[1. 编程语言常见概念与理论 6](#_Toc18338)

[1.1. 1.1.solid原则,,1.2.变量6属性,,1.3.三大流程,,1.4.异常控制流程,,1.5.流程图（Flow Chart,,1.6.多分支结构,,1.7.表达式（中缀前缀表达式,,1.8.运算符,,1.9.数据类型,,1.10.Lambda 表达式 ,,1.11.方法引用,,1.12.生命周期,,1.13.Cohesion,,1.14.耦合Coupling,,1.15.过程,,1.16.函数 ,,1.17.返回值,,1.18.循环(loop),,,1.19. 递归(recursion), ,,1.20.遍历(traversal), ,,1.21.迭代(iterate).,,1.22.泛型,,1.23.环境变量,,1.24.s-exp,,1.25.类型系统 6](#_Toc30822)

[2. 编译原理 7](#_Toc3871)

[3. 词法分析 7](#_Toc23894)

[4. Ast构建，语法分析 7](#_Toc21306)

[5. 语意分析 7](#_Toc18658)

[6. 6 数据结构 7](#_Toc14111)

[6.1. ▪ 记号 7](#_Toc16185)

[6.2. ▪ 语法树 7](#_Toc11564)

[6.3. ▪ 符号表 7](#_Toc5250)

[6.4. ▪ 常数表 7](#_Toc16162)

[6.5. ▪ 中间代码 7](#_Toc204)

[6.6. ▪ 临时文件 7](#_Toc3295)

[7. 7 其他问题 7](#_Toc29790)

[7.1. ▪ 分析和综合 7](#_Toc24043)

[7.2. ▪ 前端和后端 7](#_Toc14229)

[7.3. ▪ 遍 8](#_Toc17497)

[7.4. ▪ 语言定义和编译器 8](#_Toc31414)

[8. 1.3 程序设计语言的发展历程 8](#_Toc976)

[1.3.1 走向高级程序设计语言 　　1.3.2 对编译器的影响 8](#_Toc23908)

[9. 第6章 中间代码生成 8](#_Toc17843)

[10. 第7章 运行时刻环境 8](#_Toc29329)

[11. 编程语言种类 and趋势 逻辑式语言..函数式语言...命令式语言 8](#_Toc11208)

[12. 语言设计与实现 的概念 10](#_Toc11553)

[12.1. 3.1.词法分析,,3.2.fsm状态机,,3.3.正则表达式,,3.4.词性标注,,3.5.语法分析,,3.6.ast,,3.7.语义分析,,3.8.内部DSL实现模式,,3.9.外部DSL,,3.10.生成AST,,3.11.BNF,,3.12.正则表达式表的词法分析器,,3.13.递归下降法语法解析器,,3.14.解释器,,3.15.表达式生成器,,3.16.嵌套函数,,3.17.方法级联,,3.18.状态机fsm 10](#_Toc7370)

[13. 函数式理论 章节 10](#_Toc17059)

[13.1. 4.1.lambda,,4.2.stream ,,4.3.fluce接口,,4.4.连锁接口,,4.5.pipe,,4.6.map & reduce ,,4.7.pipeline,,4.8.高阶函数,,4.9.闭包,,4.10.表达式特化,,4.11.表达式合成,,4.12.外循环,,4.13.内循环,,4.14.惰性求值,,尾递归,,yield,,geneor,,一等函数,,闭包或者 仿函数（functor）对,,惰性计算（lazy evaluation,,递归,,只用"表达式,,尾递归，伪递归,,first-class types,,generic types,, pattern match,,Guard,,curry,,immutable 、mutable ,,引用透明（Referential transparency）的和没有副作用（No Side Effect）。,,continuation 10](#_Toc17094)

[14. 类与对象实现 章节 11](#_Toc15559)

[14.1. 5.1.class模板,,5.2.dock type,,5.3.is-a has-a,,5.4.mix in,,5.5.oo三大特性,,5.6.solid原则,,5.7.抽象方法,,5.8.抽象类和,,5.9.单根继承,,5.10.单例,,5.11.动态绑定（dynamic binding,,5.12.动态对象,,5.13.对象持久性（object persistence,,对象实现,,多继承,,多态,,反射api加载,,方法重载,,访问方法（accessor method）和,,封装,,覆盖（Overriding,,构造方法,,构造函数,,环境变量,,基于原型,,继承,,接口用interface,,类加载,,菱形继承,,魔术方法,,魔术属性,,设置方法(mutator method),,实例,,属性读写,,外部文件加载,,析勾方法,,消息,,消息传递,,延迟绑定方法,,自动语句加载,,字节数组加载 11](#_Toc12600)

[15. 章节 11](#_Toc30580)

[16. dsl 章节 11](#_Toc6469)

[17. Vm虚拟机 11](#_Toc30897)

[18. io输入输出 章节 11](#_Toc25731)

[18.1. 8.1.文件的io,,8.2.网络的io,,8.3.rest接口,,8.4.http,,8.5.socket,,8.6.console重写,,8.7.异常打印string,,8.8.字符流,,8.9.字节流,,8.10.io体系,,8.11.图片的io,,8.12.二进制io,,8.13.文本io,,8.14.字符编码,,8.15.request,,response,,stream 12](#_Toc14527)

[19. 并发与锁机制 章节 12](#_Toc4900)

[19.1. 9.1. Volatile 内存屏障,,9.2.cas ,,9.3.atomic,,9.4.同步关键字,,9.5.ReentrantLock ,,9.6.自旋锁 ,,9.7.信号量,,9.8.互斥锁Mutex,,9.9.线程死锁,,9.10.线程死锁kill,,9.11.异步,,9.12.线程池,,9.13.task模型,,9.14.futuretask模式,,9.15.毒丸,,9.16.threadlocalhost 12](#_Toc25621)

[20. api设计 章节 12](#_Toc24723)

[20.1. 方法连 12](#_Toc11800)

[20.2. jquery 12](#_Toc2485)

[20.3. guava 12](#_Toc17102)

[20.4. option 13](#_Toc10858)

[20.5. 面向接口编程 13](#_Toc5842)

[20.6. 工厂方法优于构造函数 13](#_Toc5137)

[20.7. polyfill 13](#_Toc4190)

[20.8. dsl命名法 13](#_Toc19727)

[20.9. 参考知名api 13](#_Toc6432)

[20.10. 归一化 13](#_Toc10670)

[20.11. 命名参数 13](#_Toc26675)

[21. gc资源释放 章节 15](#_Toc12491)

[21.1. 分代 15](#_Toc3194)

[21.2. 基于时间 15](#_Toc21299)

[21.3. 引用计数 15](#_Toc17193)

[21.4. amr 15](#_Toc29336)

[21.5. watchdog 15](#_Toc17219)

[21.6. 15](#_Toc23946)

[22. 调试技术 章节 15](#_Toc11857)

[22.1. log调试 15](#_Toc20439)

[22.2. ide调试 15](#_Toc6056)

[22.3. 监测工具 15](#_Toc17207)

[22.4. sdk调试支持api 15](#_Toc24305)

[23. 反射 章节 16](#_Toc31297)

[24. 异常处理 章节 17](#_Toc16331)

[24.1. 14.1.异常捕获策略,,14.2.处理策略,,14.3.抛出策略,,14.4.跨语言异常,,14.5.异常转换,,14.6.异常序列化,,14.7.Atitit 异常and 流程控制,,14.8.返回码模式,,14.9.终止模式 vs 恢复模式,,14.10.异常机制的设计原理,,14.11.类库实现异常catch,,14.12. Check ex vs unchk ex,,14.13.业务异常,,14.14. 异常分类,,14.15.重新抛出异常和异常链,,14.16.全局异常捕获,,分布式系统的异常处理,, 异常传输 17](#_Toc16361)

[25. 事件机制 章节 17](#_Toc18379)

[25.1. 15.1.event对象,,15.2.事件目标（event target,,15.3.事件传播（event propagation,,15.4.event handler,,15.5.事件分发器,,15.6.事件委托,,15.7.事件代理,,15.8.事件循环,,15.9.事件驱动 vs 轮询机制,,15.10.事件队列,,15.11.vs消息机制,,15.12.事件注册,,15.13.事件调用,,15.14.事件对象模型,,15.15.事件冒泡,,15.16.事件捕获,,15.17.事件底层 中断原理 17](#_Toc26656)

[26. 重构 章节 17](#_Toc16975)

[27. 章节 17](#_Toc5026)

[28. 开发规范 章节 18](#_Toc29887)

[29. 序列化 章节 18](#_Toc31512)

[29.1. xml序列化 18](#_Toc12333)

[29.2. php hash序列化 18](#_Toc10491)

[29.3. json序列化 18](#_Toc25887)

[29.4. 二进制序列化 18](#_Toc12542)

[29.5. 循环引用解决 18](#_Toc12463)

[29.6. base64 18](#_Toc6945)

[29.7. bin2txt 18](#_Toc9451)

[30. ioc 章节 19](#_Toc23326)

[31. 性能提升 章节 19](#_Toc28925)

[32. ----------常见api部分 19](#_Toc11371)

[32.1. 第12章  图形图像处理技术 19](#_Toc20398)

[32.2. 第13章  文件系统 19](#_Toc16172)

[32.3. 第6章  正则表达式 19](#_Toc5903)

[32.4. Web 日期时间 Cookie与Session xml 19](#_Toc17857)

[33. Ui与gui 章节 19](#_Toc23108)

[33.1. 22.1.gui原理,,22.2.界面引擎,,22.3.游戏引擎,,22.4.布局引擎,,22.5.控件系统,,22.6.复合控件,,22.7.pagging翻页控件,,22.8.css框架,,22.9.前端ui框架,,22.10.css3,,22.11.动画,,22.12.canvas,,22.13.音视频 mvc 章节 19](#_Toc12259)

[34. other 章节 20](#_Toc22211)

[35. ide 章节 20](#_Toc1885)

[36. exop互操作接口 章节 20](#_Toc28815)

[37. 常用协议 章节 20](#_Toc17388)

[38. 常用web服务器 章节 20](#_Toc23853)

[39. 代码管理与同步 章节 20](#_Toc22223)

[40. 安全与加密 章节 20](#_Toc23403)

[40.1. 29.1.rsa,,29.2.证书,,29.3.md5,,29.4.aes,,29.5.3des,,29.6.注入,,29.7.crsf,,29.8.xss 20](#_Toc29698)

[41. 架构知识 章节 20](#_Toc1582)

[42. 文字处理 章节 21](#_Toc3652)

[42.1. 汉字处理 21](#_Toc739)

[42.2. 正则表达式 21](#_Toc25684)

[42.3. auth 21](#_Toc4168)

[42.4. 全文检索 21](#_Toc12551)

[42.5. tie树 21](#_Toc24741)

[42.6. 规则查询 21](#_Toc23850)

[42.7. 参见图像处理知识图谱 21](#_Toc17072)

[43. 图像处理 章节 21](#_Toc18760)

[44. uml 章节 22](#_Toc30387)

[45. web体系 章节 22](#_Toc14028)

[46. 简单的图像处理 章节 22](#_Toc14757)

[47. vm 章节 22](#_Toc21277)

[48. api标准化 章节 22](#_Toc7772)

[48.1. 37.1.核心core标准化,,37.2.文本处理,,37.3.网络处理,,37.4.图像处理,,37.5.数据库处理,,37.6.进程处理,,37.7.鼠标键盘管理,,37.8.时间处理,,37.9.数字处理,,37.10.容器 22](#_Toc15520)

[49. 常用算法 章节 22](#_Toc19713)

[50. 软件工程 章节 22](#_Toc22726)

[51. 设计模式 章节 22](#_Toc17259)

[52. 标准化 章节 23](#_Toc19346)

[52.1. 41.1.ecma标准,,41.2.iso标准,,41.3.apache fund,,41.4.jcp jsr,,41.5.gbk,,41.6.rfc,,41.7.w3c,,41.8.ansi,,41.9.omg,,41.10.BCP，即Best Current Practice 23](#_Toc17678)

[53. 字符编码 章节 23](#_Toc9269)

[53.1. 42.1.gbk,,42.2.utf8,,42.3.unicode,,42.4.大头端小头,,42.5.ansi,,42.6.ascii,,42.7.utf bom,,42.8.bigend,,42.9.littleend,,42.10.cjk,,42.11.charset,,42.12.base64,,42.13.mime 23](#_Toc24478)

[54. bug管理 章节 23](#_Toc12216)

[55. 编程原则准则 章节 24](#_Toc20890)

[55.1. S.O.L.I.D原则(oo fp) 3 24](#_Toc10702)

[55.2. 3.1. Single Responsibility Principle (SRP)–职责单一原则 3 24](#_Toc19033)

[55.3. 3.2. Open/Closed Principle (OCP)–开闭原则 3 24](#_Toc1334)

[55.4. 3.3. Liskov substitution principle (LSP)–里氏代换原则 3 24](#_Toc30542)

[55.5. 3.4. Interface Segregation Principle (ISP)–接口隔离原则 3 24](#_Toc4203)

[55.6. 3.5. Dependency Inversion Principle(DIP)–依赖倒置原则 3 26](#_Toc12444)

[55.7. 4. Dky法则 dry kiss yagni 3 26](#_Toc8288)

[55.8. 5. Other原则 4 26](#_Toc26427)

[55.9. 5.1. 可读性优先原则 4 26](#_Toc31395)

[55.10. 5.2. 拥抱变化原则 扩展性 4 26](#_Toc18739)

[55.11. 5.3. ：迪米特法则最少知识原则”（Principle of Least Knowledge） 4 26](#_Toc29878)

[55.12. 5.4. 迪米特法则（Law?ofDemeter,?LOD） 4 26](#_Toc30028)

[55.13. 5.5. Don’t Repeat Yourself(DRY) 4 28](#_Toc6444)

[55.14. 5.6. Keep It Simple, Stupid (KISS) 4 28](#_Toc5526)

[55.15. 5.7. Program to an interface, not an implementation4 28](#_Toc7526)

[55.16. 5.8. You Ain’t Gonna Need It(YAGNI) 4 28](#_Toc12094)

[55.17. 5.9. 最小惊讶原则 遵守标准约定注释说什么代码就做什么 5 28](#_Toc23694)

[55.18. 5.10. 最小化耦合原则与最大内聚 5 28](#_Toc20485)

[55.19. 5.11. 隐藏实现细节原则 5 30](#_Toc22427)

[55.20. 5.12. 避免过早优化原则 5 30](#_Toc24154)

[55.21. 5.13. 得墨忒耳定律 5 30](#_Toc26440)

[55.22. 5.14. Common Closure Principle（CCP）–共同封闭原则 5 30](#_Toc21485)

[55.23. 5.15. Common Reuse Principle (CRP)–共同重用原则 5 30](#_Toc6881)

[55.24. 5.16. Hollywood Principle–好莱坞原则 5 30](#_Toc30155)

[55.25. 5.17. High Cohesion & Low/Loose coupling & –高内聚，低耦合6 31](#_Toc4712)

[55.26. 5.18. Convention over Configuration（CoC）–惯例优于配置原则 惯例优先原则6 31](#_Toc15989)

[55.27. 5.19. Acyclic Dependencies Principle (ADP)–无环依赖原则 6 31](#_Toc22230)

[55.28. 5.20. Postel's law伯斯塔尔法则 后向兼容性设计 6 31](#_Toc4492)

[55.29. 5.21. 软件，就像所有技术一样，具有天然的政治性。代码必然会反映作者的选择、偏见和期望。 6 31](#_Toc2445)

[55.30. 5.22. 首先为人类设计，其次为机器设计。 6 32](#_Toc19980)

[55.31. 5.23. 抽象原则 6 32](#_Toc19512)

[55.32. 6. 《编程法则》二十一条 6 32](#_Toc15146)

[56. datastruts数据结构 章节 33](#_Toc3231)

[56.1. 45.1.map,,45.2.list,,45.3.元祖,,45.4.turl,,45.5.meta、,,45.6.dic,,45.7.hashtable,,45.8.hash,,45.9.set,,45.10.array,,45.11.graph,,45.12.linklist,,45.13.tree,,45.14.collect、,,45.15.queuo,,45.16.stack,,45.17.heap,,45.18.sort,,45.19.查找搜索,,45.20.遍历,,45.21.fifo,,45.22.,, 33](#_Toc27984)

[57. 源码组织管理 章节 33](#_Toc31810)

[58. 标准库 章节 33](#_Toc27596)

[58.1. 字符串函数 34](#_Toc31165)

[58.2. 数学函数库 34](#_Toc25902)

[58.3. 日期运算 34](#_Toc8761)

[58.4. 聚合函数 34](#_Toc28528)

[58.5. 转换函数 34](#_Toc13175)

[58.6. 字符函数 34](#_Toc31307)

[58.7. 系统函数 34](#_Toc27458)

[58.8. 分析函数（高级统计函数 34](#_Toc3447)

[58.9. 条件判断函数 34](#_Toc14142)

[58.10. 加密函数 35](#_Toc31139)

[58.11. 格式化函数 35](#_Toc24694)

[59. 参考资料 35](#_Toc6756)

[59.1. 《编译原理（第2版）——计算机科学丛书》(（美）阿霍...)【 第1章 引论 35](#_Toc30744)

[59.2. atititprgrm tech tree编程语言知识点体系 v7 qb20.xlsx 35](#_Toc30833)

# 编程语言常见概念与理论

## 1.1.solid原则,,1.2.变量6属性,,1.3.三大流程,,1.4.异常控制流程,,1.5.流程图（Flow Chart,,1.6.多分支结构,,1.7.表达式（中缀前缀表达式,,1.8.运算符,,1.9.数据类型,,1.10.Lambda 表达式 ,,1.11.方法引用,,1.12.生命周期,,1.13.Cohesion,,1.14.耦合Coupling,,1.15.过程,,1.16.函数 ,,1.17.返回值,,1.18.循环(loop),,,1.19. 递归(recursion), ,,1.20.遍历(traversal), ,,1.21.迭代(iterate).,,1.22.泛型,,1.23.环境变量,,1.24.s-exp,,1.25.类型系统

# 函数

1. 6.1　用户自定义函数  
   6.1.1　语法解析  
   6.1.2　抽象语法树的编译  
   6.2　内部函数  
   6.3　函数的调用  
   6.4　函数的执行

# 命名空间

1. 8.1　概述  
   8.2　命名空间的定义  
   8.3　命名空间的使用  
   8.3.1　use导入  
   8.3.2　动态用法  
   8.4　小结

# 第3章　数据类型 变量 数组 引用 类型转换

3.1　变量  
3.1.1　变量类型  
3.1.2　内部实现  
3.2　字符串  
3.3　数组  
3.3.1　基本实现  
3.3.2　散列函数  
3.3.3　数组的初始化  
3.3.4　插入  
3.3.5　哈希冲突  
3.3.6　查找  
3.3.7　扩容  
3.4　引用  
3.5　类型转换  
3.5.1　转换为NULL  
3.5.2　转换为布尔型  
3.5.3　转换为整型  
3.5.4　转换为浮点型  
3.5.5　转换为字符串  
3.5.6　转换为数组  
3.5.7　转换为对象  
3.6　小结

# 基础语法的实现

9.1　静态变量  
9.2　常量  
9.2.1　const  
9.2.2　define()  
9.3　全局变量  
9.3.1　全局变量符号表  
9.3.2　全局变量的访问  
9.3.3　全局变量的销毁  
9.3.4　超全局变量  
9.4　分支结构  
9.4.1　if  
9.4.2　switch  
9.5　循环结构  
9.5.1　while  
9.5.2　do while  
9.5.3　for  
9.5.4　foreach  
9.6　中断及跳转  
9.6.1　break/continue  
9.6.2　goto  
9.7　include/require  
9.8　异常处理  
9.8.1　PHP中的try catch  
9.8.2　内核中的异常处理  
9.9　break/continue LABEL语法的

# 编译原理

# 词法分析

# Ast构建，语法分析

# 语意分析

# 6 [数据结构](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "6)

## ▪ [记号](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "6_1)

## ▪ [语法树](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "6_2)

## ▪ [符号表](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "6_3)

## ▪ [常数表](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "6_4)

## ▪ [中间代码](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "6_5)

## ▪ [临时文件](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "6_6)

# 7 [其他问题](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "7)

## ▪ [分析和综合](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "7_1)

## ▪ [前端和后端](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "7_2)

## ▪ [遍](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "7_3)

## ▪ [语言定义和编译器](http://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%8E%9F%E7%90%86/4194" \l "7_4)

# 1.3 程序设计语言的发展历程

　　1.3.1 走向高级程序设计语言  
　　1.3.2 对编译器的影响

# 第6章 中间代码生成

# 第7章 运行时刻环境

# 编程语言种类 and趋势 逻辑式语言..函数式语言...命令式语言

Atitit.编程语言的主要的种类and趋势 逻辑式语言..函数式语言...命令式语言p825

1. 编程语言的主要的种类 逻辑式语言..函数式语言...命令式语言 1

2. 命令式语言主要组成（运算符与控制结构，if，foreach） 2

3. 函数式语言（全部是函数） 2

4. 逻辑式语言,,不必考虑实现过程而只需考虑定义和结果 2

1. 控制结构将消失,select ,foreach 等...表格将取代select结构 3

5. 第五代语言 3

3.编程语言的主要的种类 逻辑式语言..函数式语言...命令式语言

在FAQ3里说过，世界上有C和LISP两种编程语言，你现在学了C，以后学了C++、Java、.NET

等等，也仍然只认识了半个世界。LISP和 Haskell构成了另外半个世

4.命令式语言主要组成（运算符与控制结构，if，foreach）

5.函数式语言（全部是函数）

6.逻辑式语言,,不必考虑实现过程而只需考虑定义和结果

例如著名的 prolog。逻辑式语言一般依据成熟的逻辑体系来设计，比如 first-order logic（用于 prolog），因为这样有几点好处：

1）可以进行严密的逻辑推理；

2）可以方便有效地定义 knowledge base，这在专家系统中很重要；

3）开发者不必考虑实现过程而只需考虑定义和结果。这符合专家系统的设计理念。

作者:: 老哇的爪子 Attilax 艾龙，  EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

6.1.控制结构将消失,select ,foreach 等...表格将取代select结构

只需考虑定义和结果,表格将取代select结构...

7.第五代语言

第五代语言就是自然语言又被称为知识库语言或人工智能语言，人工智能语言主要有LISP、Prolog

人们可能会问，用人工智能语言解决问题与传统的方法有什么区别呢？

传统方法通常把问题的全部知识以各种的模型表达在固定程序中，问题的求解完全在程序制导下按着预先安排好的步骤一步一步（逐条）执行。解决问题的思路与冯.诺依曼式计算机结构相吻合。当前大型数据库法、数学模型法、统计方法等都是严格结构化的方法。

对于人工智能技术要解决的问题，往往无法把全部知识都体现在固定的程序中。通常需要建立一个知识库（包含事实和推理规则），程序根据 环境和所给的输入信息以及所要解决的问题来决定自己的行动，所以它是在环境模式的制导下的推理过程。这种方法有极大的灵活性、对话能力、有自我解释能力和 学习能力。这种方法对解决一些条件和目标不大明确或不完备，（即不能很好地形式化，不好描述）的非结构化问题比传统方法好，它通常采用启发式、试探法策略 来解决问题。

其实，每种编程语言的设计都有

独到之处，体现了每种语言的精髓，在融汇百家之后积累下来的正是方法论。

# 语言设计与实现 的概念

## 3.1.词法分析,,3.2.fsm状态机,,3.3.正则表达式,,3.4.词性标注,,3.5.语法分析,,3.6.ast,,3.7.语义分析,,3.8.内部DSL实现模式,,3.9.外部DSL,,3.10.生成AST,,3.11.BNF,,3.12.正则表达式表的词法分析器,,3.13.递归下降法语法解析器,,3.14.解释器,,3.15.表达式生成器,,3.16.嵌套函数,,3.17.方法级联,,3.18.状态机fsm

# 函数式理论 章节

## 4.1.lambda,,4.2.stream ,,4.3.fluce接口,,4.4.连锁接口,,4.5.pipe,,4.6.map & reduce ,,4.7.pipeline,,4.8.高阶函数,,4.9.闭包,,4.10.表达式特化,,4.11.表达式合成,,4.12.外循环,,4.13.内循环,,4.14.惰性求值,,尾递归,,yield,,geneor,,一等函数,,闭包或者 仿函数（functor）对,,惰性计算（lazy evaluation,,递归,,只用"表达式,,尾递归，伪递归,,first-class types,,generic types,, pattern match,,Guard,,curry,,immutable 、mutable ,,引用透明（Referential transparency）的和没有副作用（No Side Effect）。,,continuation

章节

# 类与对象实现 章节

## 5.1.class模板,,5.2.dock type,,5.3.is-a has-a,,5.4.mix in,,5.5.oo三大特性,,5.6.solid原则,,5.7.抽象方法,,5.8.抽象类和,,5.9.单根继承,,5.10.单例,,5.11.动态绑定（dynamic binding,,5.12.动态对象,,5.13.对象持久性（object persistence,,对象实现,,多继承,,多态,,反射api加载,,方法重载,,访问方法（accessor method）和,,封装,,覆盖（Overriding,,构造方法,,构造函数,,环境变量,,基于原型,,继承,,接口用interface,,类加载,,菱形继承,,魔术方法,,魔术属性,,设置方法(mutator method),,实例,,属性读写,,外部文件加载,,析勾方法,,消息,,消息传递,,延迟绑定方法,,自动语句加载,,字节数组加载

# 章节

# dsl 章节

# Vm虚拟机

# io输入输出 章节

## 8.1.文件的io,,8.2.网络的io,,8.3.rest接口,,8.4.http,,8.5.socket,,8.6.console重写,,8.7.异常打印string,,8.8.字符流,,8.9.字节流,,8.10.io体系,,8.11.图片的io,,8.12.二进制io,,8.13.文本io,,8.14.字符编码,,8.15.request,,response,,stream

# 并发与锁机制 章节

## 9.1. Volatile 内存屏障,,9.2.cas ,,9.3.atomic,,9.4.同步关键字,,9.5.ReentrantLock ,,9.6.自旋锁 ,,9.7.信号量,,9.8.互斥锁Mutex,,9.9.线程死锁,,9.10.线程死锁kill,,9.11.异步,,9.12.线程池,,9.13.task模型,,9.14.futuretask模式,,9.15.毒丸,,9.16.threadlocalhost

# api设计 章节

|  |
| --- |
| 方法连 |
| jquery |
| guava |
| option |
| 面向接口编程 |
| 工厂方法优于构造函数 |
| polyfill |
| dsl命名法 |
| 参考知名api |
| 归一化 |
| 命名参数 |
| 属性风格API |
| **类型自动转换** |
| 通用属性方法attr( like jq) |
| **事件（events）** |
| **回调（callbacks）** |
| **api扩展性** |
| **dsl化** |
| **what how分离** |
| **函数式接口** |

# gc资源释放 章节

|  |
| --- |
| 分代 |
| 基于时间 |
| 引用计数 |
| amr |
| watchdog |

# 第4章　内存管理

4.1　变量的自动GC机制  
4.1.1　引用计数  
4.1.2　写时复制  
4.1.3　回收时机  
4.2　垃圾回收  
4.2.1　回收算法  
4.2.2　具体实现  
4.3　内存池  
4.3.1　内存池的初始化  
4.3.2　内存分配  
4.3.3　系统内存分配  
4.3.4　内存释放  
4.4　线程安全  
4.4.1　TSRM的基本实现  
4.4.2　线程私有数据  
4.4.3　线程局部存储  
4.5　小结

# 调试技术 章节

|  |
| --- |
| log调试 |
| ide调试 |
| 监测工具 |
| sdk调试支持api |

# 反射 章节

章节

# 异常处理 章节

## 14.1.异常捕获策略,,14.2.处理策略,,14.3.抛出策略,,14.4.跨语言异常,,14.5.异常转换,,14.6.异常序列化,,14.7.Atitit 异常and 流程控制,,14.8.返回码模式,,14.9.终止模式 vs 恢复模式,,14.10.异常机制的设计原理,,14.11.类库实现异常catch,,14.12. Check ex vs unchk ex,,14.13.业务异常,,14.14. 异常分类,,14.15.重新抛出异常和异常链,,14.16.全局异常捕获,,分布式系统的异常处理,, 异常传输

# 事件机制 章节

## 15.1.event对象,,15.2.事件目标（event target,,15.3.事件传播（event propagation,,15.4.event handler,,15.5.事件分发器,,15.6.事件委托,,15.7.事件代理,,15.8.事件循环,,15.9.事件驱动 vs 轮询机制,,15.10.事件队列,,15.11.vs消息机制,,15.12.事件注册,,15.13.事件调用,,15.14.事件对象模型,,15.15.事件冒泡,,15.16.事件捕获,,15.17.事件底层 中断原理

# 重构 章节

# 章节

# 开发规范 章节

# 序列化 章节

章节

|  |
| --- |
| xml序列化 |
| php hash序列化 |
| json序列化 |
| 二进制序列化 |
| 循环引用解决 |
| base64 |
| bin2txt |

# ioc 章节

# 性能提升 章节

# ----------常见api部分

## 第12章  图形图像处理技术

## 第13章  文件系统

## 第6章  正则表达式

## Web 日期时间 Cookie与Session xml

# 模板引擎

6.1　代码分层的思想  
6.2　实现一个简单的模板引擎骨架  
6.2.1　搭建模板引擎基础类骨架  
6.2.2　编译类骨架  
6.2.3　测试模板引擎  
6.3　模板引擎的编译  
6.3.1　实现变量标签  
6.3.2　实现foreach标签  
6.3.3　实现if…else标签  
6.3.4　对PHP原生语法的支持  
6.4　完善模板引擎  
6.4.1　模板缓存机制的实现  
6.4.2　调试和缓存清理  
6.4.3　如何使用模板  
6.5　常用模板引擎  
6.5.1　Discuz模板引擎  
6.5.2　Smarty模板引擎  
6.5.3　DedeCms模板引擎  
6.5.4　Blitz模板引擎  
6.5.5　模板引擎的一些思考

# 编码规范

14.1　文件格式  
14.1.1　文件标记  
14.1.2　文件和目录命名  
14.1.3　文件目录结构  
14.2　命名规范  
14.2.1　变量命名  
14.2.2　类及接口命名  
14.2.3　数据库命名  
14.2.4　习惯与约定  
14.3　注释规范  
14.3.1　程序注释  
14.3.2　文件注释  
14.3.3　类接口注释  
14.3.4　方法和函数注释  
14.3.5　标注的使用  
14.4　代码风格  
14.4.1　缩进和空格  
14.4.2　语句断行  
14.4.3　更好的习惯  
14.5　本章小结

# Ui与gui 章节

## 22.1.gui原理,,22.2.界面引擎,,22.3.游戏引擎,,22.4.布局引擎,,22.5.控件系统,,22.6.复合控件,,22.7.pagging翻页控件,,22.8.css框架,,22.9.前端ui框架,,22.10.css3,,22.11.动画,,22.12.canvas,,22.13.音视频 mvc 章节

# other 章节

# ide 章节

章节

# exop互操作接口 章节

# 常用协议 章节

章节

# 常用web服务器 章节

# 代码管理与同步 章节

# 安全与加密 章节

## 29.1.rsa,,29.2.证书,,29.3.md5,,29.4.aes,,29.5.3des,,29.6.注入,,29.7.crsf,,29.8.xss

# 架构知识 章节

章节

# 文字处理 章节

|  |
| --- |
| 汉字处理 |
| 正则表达式 |
| auth |
| 全文检索 |
| tie树 |
| 规则查询 |
| 参见图像处理知识图谱 |

# 图像处理 章节

章节

# uml 章节

# web体系 章节

章节

# 简单的图像处理 章节

# vm 章节

章节

# api标准化 章节

## 37.1.核心core标准化,,37.2.文本处理,,37.3.网络处理,,37.4.图像处理,,37.5.数据库处理,,37.6.进程处理,,37.7.鼠标键盘管理,,37.8.时间处理,,37.9.数字处理,,37.10.容器

# 常用算法 章节

# 软件工程 章节

章节

# 设计模式 章节

章节

# 标准化 章节

## 41.1.ecma标准,,41.2.iso标准,,41.3.apache fund,,41.4.jcp jsr,,41.5.gbk,,41.6.rfc,,41.7.w3c,,41.8.ansi,,41.9.omg,,41.10.BCP，即Best Current Practice

# 字符编码 章节

## 42.1.gbk,,42.2.utf8,,42.3.unicode,,42.4.大头端小头,,42.5.ansi,,42.6.ascii,,42.7.utf bom,,42.8.bigend,,42.9.littleend,,42.10.cjk,,42.11.charset,,42.12.base64,,42.13.mime

# bug管理 章节

章节

# 编程原则准则 章节

|  |
| --- |
| S.O.L.I.D原则(oo fp) 3 |
| 3.1. Single Responsibility Principle (SRP)–职责单一原则 3 |
| 3.2. Open/Closed Principle (OCP)–开闭原则 3 |
| 3.3. Liskov substitution principle (LSP)–里氏代换原则 3 |
| 3.4. Interface Segregation Principle (ISP)–接口隔离原则 3 |
| 3.5. Dependency Inversion Principle(DIP)–依赖倒置原则 3 |
| 4. Dky法则 dry kiss yagni 3 |
| 5. Other原则 4 |
| 5.1. 可读性优先原则 4 |
| 5.2. 拥抱变化原则 扩展性 4 |
| 5.3. ：迪米特法则最少知识原则”（Principle of Least Knowledge） 4 |
| 5.4. 迪米特法则（Law?ofDemeter,?LOD） 4 |
| 5.5. Don’t Repeat Yourself(DRY) 4 |
| 5.6. Keep It Simple, Stupid (KISS) 4 |
| 5.7. Program to an interface, not an implementation4 |
| 5.8. You Ain’t Gonna Need It(YAGNI) 4 |
| 5.9. 最小惊讶原则 遵守标准约定注释说什么代码就做什么 5 |
| 5.10. 最小化耦合原则与最大内聚 5 |
| 5.11. 隐藏实现细节原则 5 |
| 5.12. 避免过早优化原则 5 |
| 5.13. 得墨忒耳定律 5 |
| 5.14. Common Closure Principle（CCP）–共同封闭原则 5 |
| 5.15. Common Reuse Principle (CRP)–共同重用原则 5 |
| 5.16. Hollywood Principle–好莱坞原则 5 |
| 5.17. High Cohesion & Low/Loose coupling & –高内聚，低耦合6 |
| 5.18. Convention over Configuration（CoC）–惯例优于配置原则 惯例优先原则6 |
| 5.19. Acyclic Dependencies Principle (ADP)–无环依赖原则 6 |
| 5.20. Postel's law伯斯塔尔法则 后向兼容性设计 6 |
| 5.21. 软件，就像所有技术一样，具有天然的政治性。代码必然会反映作者的选择、偏见和期望。 6 |
| 5.22. 首先为人类设计，其次为机器设计。 6 |
| 5.23. 抽象原则 6 |
| 6. 《编程法则》二十一条 6 |

# datastruts数据结构 章节

## 45.1.map,,45.2.list,,45.3.元祖,,45.4.turl,,45.5.meta、,,45.6.dic,,45.7.hashtable,,45.8.hash,,45.9.set,,45.10.array,,45.11.graph,,45.12.linklist,,45.13.tree,,45.14.collect、,,45.15.queuo,,45.16.stack,,45.17.heap,,45.18.sort,,45.19.查找搜索,,45.20.遍历,,45.21.fifo,,45.22.,,

# 源码组织管理 章节

章节

# 标准库 章节

|  |
| --- |
| 字符串函数 |
| 数学函数库 |
| 日期运算 |
| 聚合函数 |
| [转换函数](http://edu.cnzz.cn/201510/97393534.shtml" \o ") |
| 字符函数 |
| 系统函数 |
| 分析函数（高级统计函数 |
| 条件判断函数 |
| 加密函数 |
| 格式化函数 |

# 参考资料

## 《编译原理（第2版）——计算机科学丛书》(（美）阿霍...)【 第1章 引论

参考资料

## atititprgrm tech tree编程语言知识点体系 v7 qb20.xlsx

## 《PHP 7 编程实战》(（美）Doug Bierer（道格·比尔） 著)【简介\_书评\_在线阅读】 - 当当图书.mhtml

## 《PHP7内核剖析》(秦朋 著)【简介\_书评\_在线阅读】 - 当当图书.mhtml