atitit 编程语言之道attilax著.docx

[1. 编程语言常见概念与理论 9](#_Toc22499)

[1.1. solid原则 9](#_Toc15962)

[1.2. 变量6属性 9](#_Toc18622)

[1.3. 三大流程 9](#_Toc16173)

[1.4. 异常控制流程 9](#_Toc28221)

[1.5. 流程图（Flow Chart 9](#_Toc28771)

[1.6. 多分支结构 9](#_Toc10406)

[1.7. 表达式（中缀前缀表达式 9](#_Toc15788)

[1.8. 运算符 9](#_Toc5980)

[1.9. 数据类型 9](#_Toc2419)

[1.10. Lambda 表达式 9](#_Toc6510)

[1.11. 方法引用 11](#_Toc19372)

[1.12. 生命周期 11](#_Toc3989)

[1.13. Cohesion 11](#_Toc28764)

[1.14. 耦合Coupling 11](#_Toc21296)

[1.15. 过程 11](#_Toc15149)

[1.16. 函数 11](#_Toc12791)

[1.17. 返回值 11](#_Toc24775)

[1.18. 循环(loop), 11](#_Toc31874)

[1.19. 递归(recursion), 11](#_Toc19395)

[1.20. 遍历(traversal), 11](#_Toc26073)

[1.21. 迭代(iterate). 12](#_Toc29304)

[1.22. 泛型 12](#_Toc9381)

[1.23. 环境变量 12](#_Toc3060)

[1.24. s-exp 12](#_Toc2742)

[1.25. 类型系统 12](#_Toc9683)

[2. 编程语言种类 and趋势 逻辑式语言..函数式语言...命令式语言 12](#_Toc11775)

[3. 语言实现 14](#_Toc27105)

[3.1. 词法分析 14](#_Toc22075)

[3.2. fsm状态机 14](#_Toc6559)

[3.3. 正则表达式 14](#_Toc15563)

[3.4. 词性标注 14](#_Toc14121)

[3.5. 语法分析 14](#_Toc22685)

[3.6. ast 14](#_Toc19979)

[3.7. 语义分析 14](#_Toc32272)

[3.8. 内部DSL实现模式 14](#_Toc9226)

[3.9. 外部DSL 14](#_Toc6960)

[3.10. 生成AST 14](#_Toc6382)

[3.11. BNF 15](#_Toc23247)

[3.12. 正则表达式表的词法分析器 15](#_Toc30153)

[3.13. 递归下降法语法解析器 15](#_Toc3382)

[3.14. 解释器 15](#_Toc12115)

[3.15. 表达式生成器 15](#_Toc27521)

[3.16. 嵌套函数 15](#_Toc31828)

[3.17. 方法级联 15](#_Toc13044)

[3.18. 状态机fsm 15](#_Toc26677)

[4. 函数式理论 章节 16](#_Toc16321)

[4.1. lambda 16](#_Toc2855)

[4.2. stream 16](#_Toc19212)

[4.3. fluce接口 16](#_Toc7303)

[4.4. 连锁接口 16](#_Toc20037)

[4.5. pipe 16](#_Toc31084)

[4.6. map & reduce 16](#_Toc24070)

[4.7. pipeline 16](#_Toc18552)

[4.8. 高阶函数 16](#_Toc8466)

[4.9. 闭包 16](#_Toc28006)

[4.10. 表达式特化 16](#_Toc27696)

[4.11. 表达式合成 17](#_Toc20980)

[4.12. 外循环 17](#_Toc4592)

[4.13. 内循环 17](#_Toc30479)

[4.14. 惰性求值 17](#_Toc11688)

[5. 类与对象实现 章节 19](#_Toc6254)

[5.1. class模板 19](#_Toc11036)

[5.2. dock type 19](#_Toc15872)

[5.3. is-a has-a 19](#_Toc6327)

[5.4. mix in 19](#_Toc12765)

[5.5. oo三大特性 19](#_Toc62)

[5.6. solid原则 19](#_Toc18066)

[5.7. 抽象方法 19](#_Toc8800)

[5.8. 抽象类和 19](#_Toc30980)

[5.9. 单根继承 19](#_Toc5018)

[5.10. 单例 19](#_Toc4675)

[5.11. 动态绑定（dynamic binding 19](#_Toc27961)

[5.12. 动态对象 20](#_Toc20727)

[5.13. 对象持久性（object persistence 20](#_Toc7050)

[6. 章节 22](#_Toc8027)

[7. dsl 章节 22](#_Toc23162)

[8. io输入输出 章节 22](#_Toc16859)

[8.1. 文件的io 22](#_Toc15404)

[8.2. 网络的io 22](#_Toc3023)

[8.3. rest接口 22](#_Toc12628)

[8.4. http 22](#_Toc21229)

[8.5. socket 22](#_Toc2392)

[8.6. console重写 22](#_Toc24421)

[8.7. 异常打印string 22](#_Toc20612)

[8.8. 字符流 22](#_Toc17862)

[8.9. 字节流 23](#_Toc25813)

[8.10. io体系 23](#_Toc16665)

[8.11. 图片的io 23](#_Toc7719)

[8.12. 二进制io 23](#_Toc5853)

[8.13. 文本io 23](#_Toc3276)

[8.14. 字符编码 23](#_Toc7103)

[8.15. request 23](#_Toc10704)

[9. 并发与锁机制 章节 24](#_Toc26445)

[9.1. Volatile 内存屏障 24](#_Toc28284)

[9.2. cas 24](#_Toc30187)

[9.3. atomic 24](#_Toc13528)

[9.4. 同步关键字 24](#_Toc13272)

[9.5. ReentrantLock 24](#_Toc4400)

[9.6. 自旋锁 24](#_Toc270)

[9.7. 信号量 24](#_Toc5365)

[9.8. 互斥锁Mutex 24](#_Toc32044)

[9.9. 线程死锁 24](#_Toc11630)

[9.10. 线程死锁kill 24](#_Toc9185)

[9.11. 异步 25](#_Toc9486)

[9.12. 线程池 25](#_Toc17766)

[9.13. task模型 25](#_Toc17522)

[9.14. futuretask模式 25](#_Toc1031)

[9.15. 毒丸 25](#_Toc14414)

[9.16. threadlocalhost 25](#_Toc1749)

[10. api设计 章节 26](#_Toc19419)

[10.1. 方法连 26](#_Toc21230)

[10.2. jquery 26](#_Toc17272)

[10.3. guava 26](#_Toc2048)

[10.4. option 26](#_Toc16803)

[10.5. 面向接口编程 26](#_Toc25078)

[10.6. 工厂方法优于构造函数 26](#_Toc12410)

[10.7. polyfill 26](#_Toc12522)

[10.8. dsl命名法 26](#_Toc15444)

[10.9. 参考知名api 26](#_Toc16853)

[10.10. 归一化 27](#_Toc4164)

[10.11. 命名参数 27](#_Toc17812)

[11. gc资源释放 章节 28](#_Toc19321)

[11.1. 分代 28](#_Toc28986)

[11.2. 基于时间 28](#_Toc7898)

[11.3. 引用计数 28](#_Toc10692)

[11.4. amr 28](#_Toc15443)

[11.5. watchdog 28](#_Toc24240)

[11.6. 28](#_Toc19880)

[12. 调试技术 章节 28](#_Toc4613)

[12.1. log调试 28](#_Toc5838)

[12.2. ide调试 28](#_Toc16837)

[12.3. 监测工具 28](#_Toc18455)

[12.4. sdk调试支持api 28](#_Toc17903)

[13. 反射 章节 29](#_Toc26291)

[14. 异常处理 章节 30](#_Toc3948)

[14.1. 异常捕获策略 30](#_Toc9821)

[14.2. 处理策略 30](#_Toc22440)

[14.3. 抛出策略 30](#_Toc9503)

[14.4. 跨语言异常 30](#_Toc3489)

[14.5. 异常转换 30](#_Toc17049)

[14.6. 异常序列化 30](#_Toc29867)

[14.7. Atitit 异常and 流程控制 30](#_Toc22068)

[14.8. 返回码模式 30](#_Toc19019)

[14.9. 终止模式 vs 恢复模式 32](#_Toc25631)

[14.10. 异常机制的设计原理 32](#_Toc13256)

[14.11. 类库实现异常catch 32](#_Toc12046)

[14.12. Check ex vs unchk ex 32](#_Toc31078)

[14.13. 业务异常 32](#_Toc13584)

[14.14. 异常分类 32](#_Toc1132)

[14.15. 重新抛出异常和异常链 32](#_Toc22521)

[14.16. 全局异常捕获 33](#_Toc32353)

[15. 事件机制 章节 33](#_Toc13871)

[15.1. event对象 34](#_Toc8995)

[15.2. 事件目标（event target 34](#_Toc7905)

[15.3. 事件传播（event propagation 34](#_Toc9413)

[15.4. event handler 34](#_Toc9455)

[15.5. 事件分发器 34](#_Toc14171)

[15.6. 事件委托 34](#_Toc11786)

[15.7. 事件代理 34](#_Toc19480)

[15.8. 事件循环 34](#_Toc2001)

[15.9. 事件驱动 vs 轮询机制 34](#_Toc32363)

[15.10. 事件队列 35](#_Toc6913)

[15.11. vs消息机制 35](#_Toc2817)

[15.12. 事件注册 35](#_Toc21740)

[15.13. 事件调用 35](#_Toc10196)

[15.14. 事件对象模型 35](#_Toc25042)

[15.15. 事件冒泡 35](#_Toc3501)

[15.16. 事件捕获 35](#_Toc26317)

[15.17. 事件底层 中断原理 35](#_Toc18118)

[16. 重构 章节 36](#_Toc8315)

[17. 章节 36](#_Toc13168)

[18. 开发规范 章节 36](#_Toc13186)

[19. 序列化 章节 36](#_Toc31974)

[19.1. xml序列化 37](#_Toc9889)

[19.2. php hash序列化 37](#_Toc12208)

[19.3. json序列化 37](#_Toc31218)

[19.4. 二进制序列化 37](#_Toc16974)

[19.5. 循环引用解决 37](#_Toc11650)

[19.6. base64 37](#_Toc12799)

[19.7. bin2txt 37](#_Toc22896)

[20. ioc 章节 37](#_Toc28373)

[21. 性能提升 章节 37](#_Toc19499)

[22. gui 章节 38](#_Toc22919)

[22.1. gui原理 38](#_Toc1935)

[22.2. 界面引擎 38](#_Toc26769)

[22.3. 游戏引擎 38](#_Toc31722)

[22.4. 布局引擎 38](#_Toc30635)

[22.5. 控件系统 38](#_Toc6199)

[22.6. 复合控件 38](#_Toc7067)

[22.7. pagging翻页控件 38](#_Toc16647)

[22.8. css框架 38](#_Toc26091)

[22.9. 前端ui框架 38](#_Toc3064)

[22.10. css3 39](#_Toc7954)

[22.11. 动画 39](#_Toc19375)

[22.12. canvas 39](#_Toc29448)

[22.13. 音视频 39](#_Toc6303)

[23. other 章节 39](#_Toc11107)

[24. mvc 章节 39](#_Toc9521)

[25. ide 章节 39](#_Toc11113)

[26. exop互操作接口 章节 39](#_Toc24583)

[27. 常用协议 章节 39](#_Toc17863)

[28. 常用web服务器 章节 40](#_Toc15725)

[29. 代码管理与同步 章节 40](#_Toc190)

[30. 安全与加密 章节 40](#_Toc2354)

[30.1. rsa 40](#_Toc17677)

[30.2. 证书 40](#_Toc27784)

[30.3. md5 40](#_Toc5347)

[30.4. aes 40](#_Toc28568)

[30.5. 3des 40](#_Toc5017)

[30.6. 注入 40](#_Toc15589)

[30.7. crsf 40](#_Toc11271)

[30.8. xss 40](#_Toc23743)

[31. 架构知识 章节 41](#_Toc21165)

[32. 文字处理 章节 41](#_Toc4433)

[32.1. 汉字处理 41](#_Toc2573)

[32.2. 正则表达式 41](#_Toc12)

[32.3. auth 41](#_Toc17127)

[32.4. 全文检索 41](#_Toc31614)

[32.5. tie树 41](#_Toc16178)

[32.6. 规则查询 41](#_Toc6361)

[32.7. 参见图像处理知识图谱 41](#_Toc5643)

[33. 图像处理 章节 42](#_Toc22195)

[34. uml 章节 42](#_Toc10225)

[35. web体系 章节 42](#_Toc1736)

[36. 简单的图像处理 章节 42](#_Toc3587)

[37. vm 章节 42](#_Toc31425)

[38. api标准化 章节 43](#_Toc9252)

[38.1. 核心core标准化 43](#_Toc13398)

[38.2. 文本处理 43](#_Toc3364)

[38.3. 网络处理 43](#_Toc6969)

[38.4. 图像处理 43](#_Toc18795)

[38.5. 数据库处理 43](#_Toc4019)

[38.6. 进程处理 43](#_Toc31667)

[38.7. 鼠标键盘管理 43](#_Toc9948)

[38.8. 时间处理 43](#_Toc5329)

[38.9. 数字处理 43](#_Toc1489)

[38.10. 容器 44](#_Toc9896)

[39. 常用算法 章节 44](#_Toc15187)

[40. 软件工程 章节 44](#_Toc24493)

[41. 设计模式 章节 44](#_Toc13242)

[42. 标准化 章节 44](#_Toc31665)

[42.1. ecma标准 45](#_Toc289)

[42.2. iso标准 45](#_Toc7111)

[42.3. apache fund 45](#_Toc16936)

[42.4. jcp jsr 45](#_Toc18042)

[42.5. gbk 45](#_Toc981)

[42.6. rfc 45](#_Toc6844)

[42.7. w3c 45](#_Toc8015)

[42.8. ansi 45](#_Toc21292)

[42.9. omg 45](#_Toc23172)

[42.10. BCP，即Best Current Practice 45](#_Toc25764)

[43. 字符编码 章节 46](#_Toc2814)

[43.1. gbk 46](#_Toc6708)

[43.2. utf8 46](#_Toc13430)

[43.3. unicode 46](#_Toc25073)

[43.4. 大头端小头 46](#_Toc10470)

[43.5. ansi 46](#_Toc22853)

[43.6. ascii 46](#_Toc7712)

[43.7. utf bom 46](#_Toc23706)

[43.8. bigend 46](#_Toc3385)

[43.9. littleend 46](#_Toc15776)

[43.10. cjk 46](#_Toc4690)

[43.11. charset 47](#_Toc7290)

[43.12. base64 47](#_Toc30584)

[43.13. mime 47](#_Toc915)

[44. bug管理 章节 47](#_Toc11186)

[45. 编程原则准则 章节 48](#_Toc8947)

[45.1. S.O.L.I.D原则(oo fp) 3 48](#_Toc12166)

[45.2. 3.1. Single Responsibility Principle (SRP)–职责单一原则 3 48](#_Toc12095)

[45.3. 3.2. Open/Closed Principle (OCP)–开闭原则 3 48](#_Toc2665)

[45.4. 3.3. Liskov substitution principle (LSP)–里氏代换原则 3 48](#_Toc18902)

[45.5. 3.4. Interface Segregation Principle (ISP)–接口隔离原则 3 48](#_Toc14592)

[45.6. 3.5. Dependency Inversion Principle(DIP)–依赖倒置原则 3 50](#_Toc17055)

[45.7. 4. Dky法则 dry kiss yagni 3 50](#_Toc16924)

[45.8. 5. Other原则 4 50](#_Toc18823)

[45.9. 5.1. 可读性优先原则 4 50](#_Toc20689)

[45.10. 5.2. 拥抱变化原则 扩展性 4 50](#_Toc5318)

[45.11. 5.3. ：迪米特法则最少知识原则”（Principle of Least Knowledge） 4 50](#_Toc25630)

[45.12. 5.4. 迪米特法则（Law?ofDemeter,?LOD） 4 50](#_Toc27228)

[45.13. 5.5. Don’t Repeat Yourself(DRY) 4 52](#_Toc26880)

[45.14. 5.6. Keep It Simple, Stupid (KISS) 4 52](#_Toc11880)

[45.15. 5.7. Program to an interface, not an implementation4 52](#_Toc31204)

[45.16. 5.8. You Ain’t Gonna Need It(YAGNI) 4 52](#_Toc25927)

[45.17. 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 52](#_Toc5837)

[45.18. 5.10. 最小化耦合原则与最大内聚 5 52](#_Toc20380)

[45.19. 5.11. 隐藏实现细节原则 5 54](#_Toc953)

[45.20. 5.12. 避免过早优化原则 5 54](#_Toc14261)

[45.21. 5.13. 得墨忒耳定律 5 54](#_Toc9981)

[45.22. 5.14. Common Closure Principle（CCP）–共同封闭原则 5 54](#_Toc736)

[45.23. 5.15. Common Reuse Principle (CRP)–共同重用原则 5 54](#_Toc20402)

[45.24. 5.16. Hollywood Principle–好莱坞原则 5 54](#_Toc28341)

[45.25. 5.17. High Cohesion & Low/Loose coupling & –高内聚，低耦合6 55](#_Toc19570)

[45.26. 5.18. Convention over Configuration（CoC）–惯例优于配置原则 惯例优先原则6 55](#_Toc24926)

[45.27. 5.19. Acyclic Dependencies Principle (ADP)–无环依赖原则 6 55](#_Toc11613)

[45.28. 5.20. Postel's law伯斯塔尔法则 后向兼容性设计 6 55](#_Toc7422)

[45.29. 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 55](#_Toc27758)

[45.30. 5.22. 首先为人类设计，其次为机器设计。 6 56](#_Toc16086)

[45.31. 5.23. 抽象原则 6 56](#_Toc29289)

[45.32. 6. 《编程法则》二十一条 6 56](#_Toc20601)

[46. datastruts数据结构 章节 57](#_Toc8283)

[46.1. map 57](#_Toc13618)

[46.2. list 57](#_Toc24870)

[46.3. 元祖 57](#_Toc30225)

[46.4. turl 57](#_Toc9322)

[46.5. meta、 57](#_Toc27584)

[46.6. dic 57](#_Toc4495)

[46.7. hashtable 57](#_Toc5090)

[46.8. hash 57](#_Toc15384)

[46.9. set 57](#_Toc22164)

[46.10. array 57](#_Toc11189)

[46.11. graph 58](#_Toc25799)

[46.12. linklist 58](#_Toc23972)

[46.13. tree 58](#_Toc24198)

[46.14. collect、 58](#_Toc21092)

[46.15. queuo 58](#_Toc10351)

[46.16. stack 58](#_Toc4557)

[46.17. heap 58](#_Toc20962)

[46.18. sort 58](#_Toc8451)

[46.19. 查找搜索 58](#_Toc14771)

[46.20. 遍历 58](#_Toc221)

[46.21. fifo 58](#_Toc19809)

[46.22. datatable 59](#_Toc16572)

[47. 源码组织管理 章节 59](#_Toc24358)

[48. 标准库 章节 59](#_Toc20710)

[48.1. 字符串函数 60](#_Toc12632)

[48.2. 数学函数库 60](#_Toc22749)

[48.3. 日期运算 60](#_Toc2542)

[48.4. 聚合函数 60](#_Toc14668)

[48.5. 转换函数 60](#_Toc30512)

[48.6. 字符函数 60](#_Toc24190)

[48.7. 系统函数 60](#_Toc30419)

[48.8. 分析函数（高级统计函数 60](#_Toc28547)

[48.9. 条件判断函数 60](#_Toc31510)

[48.10. 加密函数 61](#_Toc1893)

[48.11. 格式化函数 61](#_Toc26006)

# 编程语言常见概念与理论

|  |
| --- |
| solid原则 |
| 变量6属性 |
| 三大流程 |
| 异常控制流程 |
| 流程图（Flow Chart |
| 多分支结构 |
| 表达式（中缀前缀表达式 |
| 运算符 |
| 数据类型 |
| Lambda 表达式 |
| 方法引用 |
| 生命周期 |
| Cohesion |
| 耦合Coupling |
| 过程 |
| 函数 |
| 返回值 |
| 循环(loop), |
| 递归(recursion), |
| 遍历(traversal), |
| 迭代(iterate). |
| 泛型 |
| 环境变量 |
| s-exp |
| 类型系统 |

# 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。

# 语言实现

|  |
| --- |
| 词法分析 |
| fsm状态机 |
| 正则表达式 |
| 词性标注 |
| 语法分析 |
| ast |
| 语义分析 |
| 内部DSL实现模式 |
| 外部DSL |
| 生成AST |
| BNF |
| 正则表达式表的词法分析器 |
| 递归下降法语法解析器 |
| 解释器 |
| 表达式生成器 |
| 嵌套函数 |
| 方法级联 |
| 状态机fsm |

# 函数式理论 章节

|  |
| --- |
| lambda |
| stream |
| fluce接口 |
| 连锁接口 |
| pipe |
| map & reduce |
| pipeline |
| 高阶函数 |
| 闭包 |
| 表达式特化 |
| 表达式合成 |
| 外循环 |
| 内循环 |
| 惰性求值 |
| **尾递归** |
| yield |
| geneor |
| 一等函数 |
| [闭包或者 仿函数（functor）对](http://baike.baidu.com/view/2070037.htm" \o ") |
| 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

|  |
| --- |
| class模板 |
| dock type |
| is-a has-a |
| mix in |
| oo三大特性 |
| solid原则 |
| 抽象方法 |
| 抽象类和 |
| 单根继承 |
| 单例 |
| 动态绑定（dynamic binding |
| 动态对象 |
| 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

# io输入输出 章节

|  |
| --- |
| 文件的io |
| 网络的io |
| rest接口 |
| http |
| socket |
| console重写 |
| 异常打印string |
| 字符流 |
| 字节流 |
| io体系 |
| 图片的io |
| 二进制io |
| 文本io |
| 字符编码 |
| request |
| response |
| stream |

# 并发与锁机制 章节

|  |
| --- |
| Volatile 内存屏障 |
| cas |
| atomic |
| 同步关键字 |
| ReentrantLock |
| 自旋锁 |
| 信号量 |
| 互斥锁Mutex |
| 线程死锁 |
| 线程死锁kill |
| 异步 |
| 线程池 |
| task模型 |
| futuretask模式 |
| 毒丸 |
| 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 |

## 

# 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

|  |
| --- |
| 异常捕获策略 |
| 处理策略 |
| 抛出策略 |
| 跨语言异常 |
| 异常转换 |
| 异常序列化 |
| Atitit 异常and 流程控制 |
| 返回码模式 |
| 终止模式 vs 恢复模式 |
| 异常机制的设计原理 |
| 类库实现异常catch |
| Check ex vs unchk ex |
| 业务异常 |
| 异常分类 |
| 重新抛出异常和异常链 |
| 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

章节

|  |
| --- |
| event对象 |
| 事件目标（event target |
| 事件传播（event propagation |
| event handler |
| 事件分发器 |
| 事件委托 |
| 事件代理 |
| 事件循环 |
| 事件驱动 vs 轮询机制 |
| 事件队列 |
| vs消息机制 |
| 事件注册 |
| 事件调用 |
| 事件对象模型 |
| 事件冒泡 |
| 事件捕获 |
| 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

章节

# gui 章节

|  |
| --- |
| gui原理 |
| 界面引擎 |
| 游戏引擎 |
| 布局引擎 |
| 控件系统 |
| 复合控件 |
| pagging翻页控件 |
| css框架 |
| 前端ui框架 |
| css3 |
| 动画 |
| canvas |
| 音视频 |

# other 章节

# mvc 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

|  |
| --- |
| rsa |
| 证书 |
| md5 |
| aes |
| 3des |
| 注入 |
| crsf |
| 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

|  |
| --- |
| 核心core标准化 |
| 文本处理 |
| 网络处理 |
| 图像处理 |
| 数据库处理 |
| 进程处理 |
| 鼠标键盘管理 |
| 时间处理 |
| 数字处理 |
| 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

|  |
| --- |
| ecma标准 |
| iso标准 |
| apache fund |
| jcp jsr |
| gbk |
| rfc |
| w3c |
| ansi |
| omg |
| BCP，即Best Current Practice |

# 字符编码 章节

|  |
| --- |
| gbk |
| utf8 |
| unicode |
| 大头端小头 |
| ansi |
| ascii |
| utf bom |
| bigend |
| littleend |
| cjk |
| charset |
| base64 |
| 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

|  |
| --- |
| map |
| list |
| 元祖 |
| turl |
| meta、 |
| dic |
| hashtable |
| hash |
| set |
| array |
| graph |
| linklist |
| tree |
| collect、 |
| queuo |
| stack |
| heap |
| sort |
| 查找搜索 |
| 遍历 |
| fifo |
| datatable |

# 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

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