atitit 编程语言之道attilax著.docx

[1. 编程语言常见概念与理论 9](#_Toc14741)

[1.1. solid原则 9](#_Toc26651)

[1.2. 变量6属性 9](#_Toc18772)

[1.3. 三大流程 9](#_Toc1211)

[1.4. 异常控制流程 9](#_Toc1285)

[1.5. 流程图（Flow Chart 9](#_Toc15769)

[1.6. 多分支结构 9](#_Toc13495)

[1.7. 表达式（中缀前缀表达式 9](#_Toc17723)

[1.8. 运算符 9](#_Toc26822)

[1.9. 数据类型 9](#_Toc6864)

[1.10. Lambda 表达式 9](#_Toc6875)

[1.11. 方法引用 10](#_Toc21427)

[1.12. 生命周期 10](#_Toc19444)

[1.13. Cohesion 10](#_Toc20932)

[1.14. 耦合Coupling 10](#_Toc12229)

[1.15. 过程 10](#_Toc21984)

[1.16. 函数 10](#_Toc14310)

[1.17. 返回值 10](#_Toc5822)

[1.18. 循环(loop), 10](#_Toc9644)

[1.19. 递归(recursion), 10](#_Toc2033)

[1.20. 遍历(traversal), 10](#_Toc16294)

[1.21. 迭代(iterate). 11](#_Toc6639)

[1.22. 泛型 11](#_Toc28683)

[1.23. 环境变量 11](#_Toc17219)

[1.24. s-exp 11](#_Toc31652)

[1.25. 类型系统 11](#_Toc25685)

[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 11](#_Toc15604)

[3. 语言实现 13](#_Toc25199)

[3.1. 词法分析 13](#_Toc28075)

[3.2. fsm状态机 13](#_Toc31715)

[3.3. 正则表达式 13](#_Toc6311)

[3.4. 词性标注 13](#_Toc9862)

[3.5. 语法分析 13](#_Toc24851)

[3.6. ast 13](#_Toc4828)

[3.7. 语义分析 13](#_Toc28515)

[3.8. 内部DSL实现模式 13](#_Toc411)

[3.9. 外部DSL 13](#_Toc6969)

[3.10. 生成AST 13](#_Toc18048)

[3.11. BNF 14](#_Toc24956)

[3.12. 正则表达式表的词法分析器 14](#_Toc27266)

[3.13. 递归下降法语法解析器 14](#_Toc28716)

[3.14. 解释器 14](#_Toc12522)

[3.15. 表达式生成器 14](#_Toc7598)

[3.16. 嵌套函数 14](#_Toc7640)

[3.17. 方法级联 14](#_Toc4009)

[3.18. 状态机fsm 14](#_Toc21049)

[4. 函数式理论 章节 15](#_Toc13745)

[4.1. lambda 15](#_Toc28739)

[4.2. stream 15](#_Toc7754)

[4.3. fluce接口 15](#_Toc5358)

[4.4. 连锁接口 15](#_Toc8461)

[4.5. pipe 15](#_Toc5627)

[4.6. map & reduce 15](#_Toc28410)

[4.7. pipeline 15](#_Toc29077)

[4.8. 高阶函数 15](#_Toc25136)

[4.9. 闭包 15](#_Toc906)

[4.10. 表达式特化 15](#_Toc28844)

[4.11. 表达式合成 16](#_Toc30120)

[4.12. 外循环 16](#_Toc446)

[4.13. 内循环 16](#_Toc19408)

[4.14. 惰性求值 16](#_Toc9293)

[5. 类与对象实现 章节 18](#_Toc12720)

[5.1. class模板 18](#_Toc21544)

[5.2. dock type 18](#_Toc8009)

[5.3. is-a has-a 18](#_Toc12022)

[5.4. mix in 18](#_Toc21165)

[5.5. oo三大特性 18](#_Toc15744)

[5.6. solid原则 18](#_Toc20627)

[5.7. 抽象方法 18](#_Toc23655)

[5.8. 抽象类和 18](#_Toc27320)

[5.9. 单根继承 18](#_Toc26508)

[5.10. 单例 18](#_Toc554)

[5.11. 动态绑定（dynamic binding 18](#_Toc9649)

[5.12. 动态对象 19](#_Toc11495)

[5.13. 对象持久性（object persistence 19](#_Toc7705)

[6. 章节 21](#_Toc24951)

[7. dsl 章节 21](#_Toc4146)

[8. io输入输出 章节 21](#_Toc8305)

[8.1. 文件的io 21](#_Toc17809)

[8.2. 网络的io 21](#_Toc4873)

[8.3. rest接口 21](#_Toc30918)

[8.4. http 21](#_Toc3858)

[8.5. socket 21](#_Toc8144)

[8.6. console重写 21](#_Toc13346)

[8.7. 异常打印string 21](#_Toc7996)

[8.8. 字符流 21](#_Toc12977)

[8.9. 字节流 22](#_Toc21300)

[8.10. io体系 22](#_Toc5493)

[8.11. 图片的io 22](#_Toc1543)

[8.12. 二进制io 22](#_Toc26177)

[8.13. 文本io 22](#_Toc26611)

[8.14. 字符编码 22](#_Toc4910)

[8.15. request 22](#_Toc31286)

[9. 并发与锁机制 章节 23](#_Toc24037)

[9.1. Volatile 内存屏障 23](#_Toc12210)

[9.2. cas 23](#_Toc11827)

[9.3. atomic 23](#_Toc22439)

[9.4. 同步关键字 23](#_Toc10839)

[9.5. ReentrantLock 23](#_Toc19283)

[9.6. 自旋锁 23](#_Toc19113)

[9.7. 信号量 23](#_Toc23927)

[9.8. 互斥锁Mutex 23](#_Toc2954)

[9.9. 线程死锁 23](#_Toc17166)

[9.10. 线程死锁kill 23](#_Toc14267)

[9.11. 异步 24](#_Toc9916)

[9.12. 线程池 24](#_Toc22158)

[9.13. task模型 24](#_Toc25795)

[9.14. futuretask模式 24](#_Toc25750)

[9.15. 毒丸 24](#_Toc7031)

[9.16. threadlocalhost 24](#_Toc29365)

[10. api设计 章节 25](#_Toc23694)

[10.1. 方法连 25](#_Toc26296)

[10.2. jquery 25](#_Toc28100)

[10.3. guava 25](#_Toc12788)

[10.4. option 25](#_Toc15261)

[10.5. 面向接口编程 25](#_Toc13675)

[10.6. 工厂方法优于构造函数 25](#_Toc30548)

[10.7. polyfill 25](#_Toc2187)

[10.8. dsl命名法 25](#_Toc24894)

[10.9. 参考知名api 25](#_Toc16012)

[10.10. 归一化 26](#_Toc2845)

[10.11. 命名参数 26](#_Toc26600)

[11. gc资源释放 章节 27](#_Toc31292)

[11.1. 分代 27](#_Toc14481)

[11.2. 基于时间 27](#_Toc3920)

[11.3. 引用计数 27](#_Toc16585)

[11.4. amr 27](#_Toc2668)

[11.5. watchdog 27](#_Toc21963)

[11.6. 27](#_Toc29580)

[12. 调试技术 章节 27](#_Toc28834)

[12.1. log调试 27](#_Toc13443)

[12.2. ide调试 27](#_Toc15591)

[12.3. 监测工具 27](#_Toc22353)

[12.4. sdk调试支持api 27](#_Toc7742)

[13. 反射 章节 28](#_Toc13354)

[14. 异常处理 章节 29](#_Toc27323)

[14.1. 异常捕获策略 29](#_Toc29879)

[14.2. 处理策略 29](#_Toc18856)

[14.3. 抛出策略 29](#_Toc15957)

[14.4. 跨语言异常 29](#_Toc24896)

[14.5. 异常转换 29](#_Toc9056)

[14.6. 异常序列化 29](#_Toc4940)

[14.7. Atitit 异常and 流程控制 29](#_Toc15312)

[14.8. 返回码模式 29](#_Toc18394)

[14.9. 终止模式 vs 恢复模式 30](#_Toc1963)

[14.10. 异常机制的设计原理 30](#_Toc32346)

[14.11. 类库实现异常catch 30](#_Toc4634)

[14.12. Check ex vs unchk ex 30](#_Toc24800)

[14.13. 业务异常 30](#_Toc16938)

[14.14. 异常分类 30](#_Toc12907)

[14.15. 重新抛出异常和异常链 30](#_Toc28279)

[14.16. 全局异常捕获 31](#_Toc30808)

[15. 事件机制 章节 31](#_Toc13143)

[15.1. event对象 32](#_Toc103)

[15.2. 事件目标（event target 32](#_Toc18241)

[15.3. 事件传播（event propagation 32](#_Toc12290)

[15.4. event handler 32](#_Toc10114)

[15.5. 事件分发器 32](#_Toc11942)

[15.6. 事件委托 32](#_Toc1919)

[15.7. 事件代理 32](#_Toc19609)

[15.8. 事件循环 32](#_Toc32526)

[15.9. 事件驱动 vs 轮询机制 32](#_Toc20565)

[15.10. 事件队列 33](#_Toc6329)

[15.11. vs消息机制 33](#_Toc26709)

[15.12. 事件注册 33](#_Toc9949)

[15.13. 事件调用 33](#_Toc11292)

[15.14. 事件对象模型 33](#_Toc16138)

[15.15. 事件冒泡 33](#_Toc11956)

[15.16. 事件捕获 33](#_Toc29961)

[15.17. 事件底层 中断原理 33](#_Toc30205)

[16. 重构 章节 34](#_Toc31606)

[17. 章节 34](#_Toc1402)

[18. 开发规范 章节 34](#_Toc15829)

[19. 序列化 章节 34](#_Toc24704)

[19.1. xml序列化 35](#_Toc26750)

[19.2. php hash序列化 35](#_Toc18166)

[19.3. json序列化 35](#_Toc13539)

[19.4. 二进制序列化 35](#_Toc18701)

[19.5. 循环引用解决 35](#_Toc17650)

[19.6. base64 35](#_Toc29064)

[19.7. bin2txt 35](#_Toc28327)

[20. ioc 章节 35](#_Toc921)

[21. 性能提升 章节 35](#_Toc8455)

[22. gui 章节 36](#_Toc28768)

[22.1. gui原理 36](#_Toc5659)

[22.2. 界面引擎 36](#_Toc2437)

[22.3. 游戏引擎 36](#_Toc25001)

[22.4. 布局引擎 36](#_Toc24968)

[22.5. 控件系统 36](#_Toc10932)

[22.6. 复合控件 36](#_Toc4034)

[22.7. pagging翻页控件 36](#_Toc3602)

[22.8. css框架 36](#_Toc32037)

[22.9. 前端ui框架 36](#_Toc23189)

[22.10. css3 37](#_Toc18490)

[22.11. 动画 37](#_Toc7068)

[22.12. canvas 37](#_Toc506)

[22.13. 音视频 37](#_Toc19716)

[23. other 章节 37](#_Toc21099)

[24. mvc 章节 37](#_Toc11219)

[25. ide 章节 37](#_Toc19245)

[26. exop互操作接口 章节 37](#_Toc28954)

[27. 常用协议 章节 37](#_Toc8021)

[28. 常用web服务器 章节 38](#_Toc28879)

[29. 代码管理与同步 章节 38](#_Toc29518)

[30. 安全与加密 章节 38](#_Toc9406)

[30.1. rsa 38](#_Toc19607)

[30.2. 证书 38](#_Toc11201)

[30.3. md5 38](#_Toc10389)

[30.4. aes 38](#_Toc18388)

[30.5. 3des 38](#_Toc15061)

[30.6. 注入 38](#_Toc31179)

[30.7. crsf 38](#_Toc5739)

[30.8. xss 38](#_Toc4017)

[31. 架构知识 章节 39](#_Toc25436)

[32. 文字处理 章节 39](#_Toc25365)

[32.1. 汉字处理 39](#_Toc2092)

[32.2. 正则表达式 39](#_Toc11986)

[32.3. auth 39](#_Toc12553)

[32.4. 全文检索 39](#_Toc24605)

[32.5. tie树 39](#_Toc10540)

[32.6. 规则查询 39](#_Toc9422)

[32.7. 参见图像处理知识图谱 39](#_Toc24679)

[33. 图像处理 章节 40](#_Toc18766)

[34. uml 章节 40](#_Toc1528)

[35. web体系 章节 40](#_Toc19082)

[36. 简单的图像处理 章节 40](#_Toc11031)

[37. vm 章节 40](#_Toc25813)

[38. api标准化 章节 41](#_Toc13022)

[38.1. 核心core标准化 41](#_Toc28332)

[38.2. 文本处理 41](#_Toc17013)

[38.3. 网络处理 41](#_Toc1712)

[38.4. 图像处理 41](#_Toc15279)

[38.5. 数据库处理 41](#_Toc5714)

[38.6. 进程处理 41](#_Toc23928)

[38.7. 鼠标键盘管理 41](#_Toc30471)

[38.8. 时间处理 41](#_Toc16579)

[38.9. 数字处理 41](#_Toc13827)

[38.10. 容器 42](#_Toc24026)

[39. 常用算法 章节 42](#_Toc4800)

[40. 软件工程 章节 42](#_Toc25918)

[41. 设计模式 章节 42](#_Toc30438)

[42. 标准化 章节 42](#_Toc20601)

[42.1. ecma标准 43](#_Toc23118)

[42.2. iso标准 43](#_Toc29330)

[42.3. apache fund 43](#_Toc23601)

[42.4. jcp jsr 43](#_Toc19710)

[42.5. gbk 43](#_Toc25289)

[42.6. rfc 43](#_Toc21829)

[42.7. w3c 43](#_Toc23781)

[42.8. ansi 43](#_Toc30908)

[42.9. omg 43](#_Toc23876)

[42.10. BCP，即Best Current Practice 43](#_Toc22261)

[43. 字符编码 章节 44](#_Toc17849)

[43.1. gbk 44](#_Toc12507)

[43.2. utf8 44](#_Toc10331)

[43.3. unicode 44](#_Toc14745)

[43.4. 大头端小头 44](#_Toc30547)

[43.5. ansi 44](#_Toc8117)

[43.6. ascii 44](#_Toc8203)

[43.7. utf bom 44](#_Toc6153)

[43.8. bigend 44](#_Toc21694)

[43.9. littleend 44](#_Toc16768)

[43.10. cjk 44](#_Toc8135)

[43.11. charset 45](#_Toc12225)

[43.12. base64 45](#_Toc30233)

[43.13. mime 45](#_Toc7214)

[44. bug管理 章节 45](#_Toc27335)

[45. 编程原则准则 章节 46](#_Toc30689)

[45.1. S.O.L.I.D原则(oo fp) 3 46](#_Toc11921)

[45.2. 3.1. Single Responsibility Principle (SRP)–职责单一原则 3 46](#_Toc28324)

[45.3. 3.2. Open/Closed Principle (OCP)–开闭原则 3 46](#_Toc10617)

[45.4. 3.3. Liskov substitution principle (LSP)–里氏代换原则 3 46](#_Toc23681)

[45.5. 3.4. Interface Segregation Principle (ISP)–接口隔离原则 3 46](#_Toc23235)

[45.6. 3.5. Dependency Inversion Principle(DIP)–依赖倒置原则 3 47](#_Toc5154)

[45.7. 4. Dky法则 dry kiss yagni 3 47](#_Toc7428)

[45.8. 5. Other原则 4 47](#_Toc12048)

[45.9. 5.1. 可读性优先原则 4 47](#_Toc20421)

[45.10. 5.2. 拥抱变化原则 扩展性 4 47](#_Toc4684)

[45.11. 5.3. ：迪米特法则最少知识原则”（Principle of Least Knowledge） 4 47](#_Toc8498)

[45.12. 5.4. 迪米特法则（Law?ofDemeter,?LOD） 4 48](#_Toc25530)

[45.13. 5.5. Don’t Repeat Yourself(DRY) 4 48](#_Toc13926)

[45.14. 5.6. Keep It Simple, Stupid (KISS) 4 48](#_Toc30663)

[45.15. 5.7. Program to an interface, not an implementation4 48](#_Toc12011)

[45.16. 5.8. You Ain’t Gonna Need It(YAGNI) 4 48](#_Toc24923)

[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 48](#_Toc9574)

[45.18. 5.10. 最小化耦合原则与最大内聚 5 49](#_Toc24160)

[45.19. 5.11. 隐藏实现细节原则 5 49](#_Toc7824)

[45.20. 5.12. 避免过早优化原则 5 49](#_Toc27994)

[45.21. 5.13. 得墨忒耳定律 5 49](#_Toc8412)

[45.22. 5.14. Common Closure Principle（CCP）–共同封闭原则 5 49](#_Toc31975)

[45.23. 5.15. Common Reuse Principle (CRP)–共同重用原则 5 49](#_Toc17028)

[45.24. 5.16. Hollywood Principle–好莱坞原则 5 50](#_Toc27683)

[45.25. 5.17. High Cohesion & Low/Loose coupling & –高内聚，低耦合6 50](#_Toc10453)

[45.26. 5.18. Convention over Configuration（CoC）–惯例优于配置原则 惯例优先原则6 50](#_Toc2880)

[45.27. 5.19. Acyclic Dependencies Principle (ADP)–无环依赖原则 6 50](#_Toc2795)

[45.28. 5.20. Postel's law伯斯塔尔法则 后向兼容性设计 6 51](#_Toc1495)

[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 51](#_Toc6974)

[45.30. 5.22. 首先为人类设计，其次为机器设计。 6 51](#_Toc11447)

[45.31. 5.23. 抽象原则 6 51](#_Toc5809)

[45.32. 6. 《编程法则》二十一条 6 51](#_Toc833)

[46. datastruts数据结构 章节 52](#_Toc27908)

[46.1. map 52](#_Toc21652)

[46.2. list 52](#_Toc21966)

[46.3. 元祖 52](#_Toc30477)

[46.4. turl 52](#_Toc32377)

[46.5. meta、 52](#_Toc22767)

[46.6. dic 52](#_Toc8393)

[46.7. hashtable 52](#_Toc1844)

[46.8. hash 52](#_Toc3834)

[46.9. set 52](#_Toc10182)

[46.10. array 52](#_Toc27003)

[46.11. graph 53](#_Toc5071)

[46.12. linklist 53](#_Toc21069)

[46.13. tree 53](#_Toc26902)

[46.14. collect、 53](#_Toc28573)

[46.15. queuo 53](#_Toc30290)

[46.16. stack 53](#_Toc25221)

[46.17. heap 53](#_Toc28518)

[46.18. sort 53](#_Toc7430)

[46.19. 查找搜索 53](#_Toc32083)

[46.20. 遍历 53](#_Toc14166)

[46.21. fifo 53](#_Toc17857)

[46.22. datatable 54](#_Toc26044)

[47. 源码组织管理 章节 54](#_Toc8516)

[48. 标准库 章节 54](#_Toc25781)

[48.1. 字符串函数 55](#_Toc7924)

[48.2. 数学函数库 55](#_Toc28273)

[48.3. 日期运算 55](#_Toc8574)

[48.4. 聚合函数 55](#_Toc3898)

[48.5. 转换函数 55](#_Toc12043)

[48.6. 字符函数 55](#_Toc23418)

[48.7. 系统函数 55](#_Toc7609)

[48.8. 分析函数（高级统计函数 55](#_Toc32454)

[48.9. 条件判断函数 55](#_Toc10482)

[48.10. 加密函数 56](#_Toc13412)

[48.11. 格式化函数 56](#_Toc24159)

# 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

# 常见类库api=----------------

## 源码组织管理 章节

章节

## 标准库 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 |

## 重构 章节 章节

## 开发规范 章节

## 序列化 章节

章节

# 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

# 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

# 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 |

# 语言高级机制------------------

## 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 |

|  |
| --- |
| 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 |

# 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 |

|  |
| --- |
| 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