# 目录

# 计算机编程通用逻辑知识概念

## 简介

计算机和编程等领域内，存在很多跨语言，跨领域的通用知识，现整理出来，供参考。

## 源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

### Gitook源码

- crifan/program_common_logic: 计算机编程通用逻辑知识概念

### 在线浏览

- 计算机编程通用逻辑知识概念 book.crifan.com
- 计算机编程通用逻辑知识概念 crifan.github.io

### 离线下载阅读

- 计算机编程通用逻辑知识概念 PDF
- 计算机编程通用逻辑知识概念 ePub
- 计算机编程通用逻辑知识概念 Mobi

# 计算机编程领域

此处介绍计算机编程领域内的一些通用知识、逻辑和概念。

# 版本号命名和版本发布

## 版本号命名规范

比如每次发版本的app，如果只是小功能更新，则只是第二位加1（或根据功能数量决定），否则大的版本的改进，则是第一位加1

Configuration · GitBook Toolchain Documentation

是有个官网推荐的做法的：

Semantic Versioning 2.0.0 | Semantic Versioning

其解释是：

> Given a version number MAJOR.MINOR.PATCH, increment the:
>
> - `MAJOR version` : when you make incompatible API changes,
> - `MINOR version` : when you add functionality in a backwards-compatible manner, and
> - `PATCH version` : when you make backwards-compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

中文版翻译是：

语义化版本 2.0.0

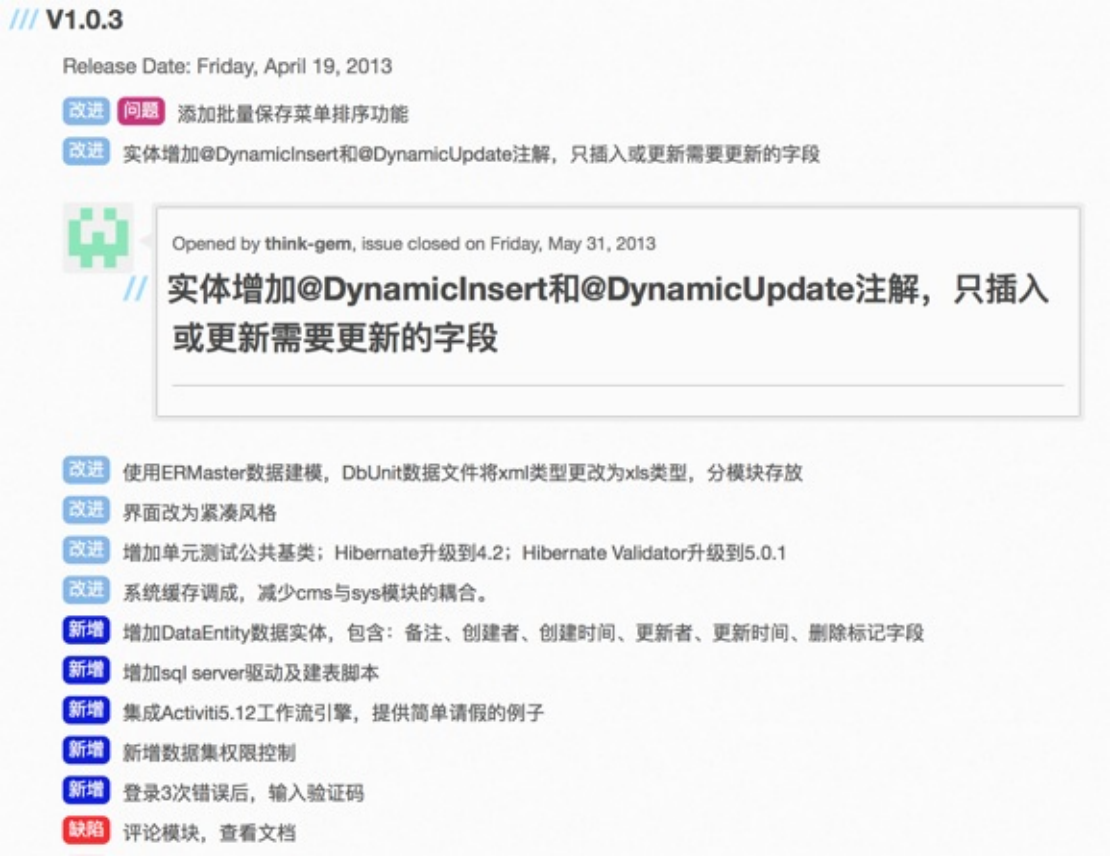
> 版本格式：主版本号.次版本号.修订号，版本号递增规则如下：
>
> 1. 主版本号：当你做了不兼容的 API 修改，
> 2. 次版本号：当你做了向下兼容的功能性新增，
> 3. 修订号：当你做了向下兼容的问题修正。
>
> 先行版本号及版本编译信息可以加到“主版本号.次版本号.修订号”的后面，作为延伸。

现在个人的常见做法是：

1. `主版本` ： 有重大功能更新
   - 比如多加了一个大的功能模块
2. `次版本` ： 有一些重要更新
   - 比如部分功能有重大优化
3. `补丁版本/小版本` ： 细节的优化
   - 比如一些小功能的优化，修复了一些小bug等等

## 版本发布的一些实践和做法

对于发布新版本，去写更新日志时，其形式可以借鉴jeesite的做法：




其效果很醒目，且点击每条可以展开显示详情

但是建议关键字还是用个人目前常用的做法：

- `新增`： 增加了新功能
- `修复`： 修复了之前存在的bug和问题
- `优化`： 改进，优化了已有的东西

# 长期支持版本LTS

`LTS` = `Long Term Support` = `长期支持`

背景

在软件领域内，很多的软件，发布的版本，随着时间的流逝，很多软件会发生很多可能，其中很大的可能是，过了一段，或许可以称为足够长的时间后，该版本的软件，已经不被原先的发布商所支持了。导致依赖于此版本的软件的其他软件，变得不可用了。这种现象，尤其是在大的软件和系统中比较常见。

解决办法

对此，大的软件发布商为了告诉其他人说，我发布的某某版本的某个软件，会长期支持，不会中断，保持一定成程度的稳定，可以被信任，则此版本的软件，往往就叫做LTS，长期支持版本。

比如：

# Ubuntu的LTS版本

Ubuntu就发布过很多个LTS的版本，比如 `Ubuntu 12.04 LTS`

详见：LTS - Ubuntu Wiki

# Node.js的LTS版本

Node.js的也有LTS版本

# 命令行终端

计算机领域内，常常会提及到**终端**，也叫做 `命令行` ，往往指的是：

- `Windows中的cmd`
- `Linux中的shell`
- `Mac中的terminal`

然后还有很多第三方的扩展、工具或插件，用于增强或替换系统自带的终端，比如：

- `Windows` 的 `Power Shell`
- `Linux` / `Mac` 的 `zsh`
- `Mac` 的 `iTerm`

# 终端能用来做什么

答 运行各种脚本和命令，完成各种任务

而对于linux的 `shell` 来说，功能强大到都算作一门单独的语言：`shell脚本语言`

可以利用不同的语法和函数，编写复杂逻辑的代码，实现特定的功能

比如小功能：

自己的批量重命名的shell脚本

TODO: 把脚本找到 贴出来

比如，其他人用shell去给OpenWrt的极路由去写应用：

开发极路由云插件 - OpenWrt.io

比如：

在linux mac win 下运行我的python脚本BlogsToWordpress

用vi在无图形界面的centos中编辑文本文件

# 各种终端的用法 心得

下面来介绍各种不同的终端的各种用法和心得。

## `Windows` 中的终端

### `Windows` 自带的 `cmd`

TODO: 把之前那个cmd心得的帖子整理过来

### `Windows` 中的 `Hyper Terminal`

嵌入式中都有和终端有关的工具使用

嵌入式开发期间，会遇到把开发版中运行程序的输出打印显示出来，以及输入和命令执行，也需要用到终端。

有时候Windows自带的terminal也够用，但是Win7中有个更好用一点的：Hyper Terminal

详见：【整理】如何在Win7中安装使用超级终端Hyper Terminal – 在路上

### Mac 的中的终端

### Mac 中自带的 terminal

### Mac 中的 iTerm2

TODO: 整理mac的iterm2的好用之处

### Linux 的 shell

linux shell编程

## shell入门

1. 快速有个了解和概念：
   - Shell脚本编程30分钟入门
2. 完整和深入的学习
   - Shell 教程 | 菜鸟教程
   - Linux Shell脚本教程：30分钟玩转Shell脚本编程_Shell中文网
3. 其它还可以的资料
   - LINUX与UNIX Shell编程指南
   - Shell编程基础 - Ubuntu中文
   - 详细介绍Linux shell脚本基础学习(一) - 雪洁 - 博客园
4. 英文不错的话，直接看英文
   - 比较系统的英文教程
     - Advanced Bash-Scripting Guide
     - BashGuide - Greg's Wiki
   - 其它更加全面的，更加系统的书籍
     - 《UNIX Shells by Example Fourth Edition》
       - chm版本下载：Prentice.Hall-Unix.Shells.By.Example,4th.Edition.chm_微盘下载
       - Linux Command Line and Shell Scripting Bible第三版
5. 对于Shell有了基本的了解和实践之后，再去看
   - Shell脚本编程的常识
     - 这些往往是经常用到，但是各种网络上的材料都语焉不详的东西，个人认为比较有用）

## 支持多平台的 SecureCRT

当然，远程操作（CentOS等）Linux类系统，也是通过终端类等工具去操作的

比如常用的SecureCRT

TODO：把 【crifan推荐】支持多种协议的串口开发工具：SecureCRT

【crifan推荐】极佳的串口开发工具：SecureCRT – 在路上

整理过来。

## 其他环境中的终端

## IDE工具中集成的终端

其他第三方工具，尤其是IDE，为了方便开发同时集成了终端

比如 VSCode PyCharm

详见 TODO： ide总结的帖子

## `Android` 的app去模拟终端

【已解决】android中的shell命令行工具 – 在路上

以便于我们在手机中通过终端去操作系统的资源，实现各种复杂的功能

# Help查看帮助信息

用help查看帮助信息

用help获取主命令和子命令的帮助信息

想要搞懂命令的基本语法，可以用：

`some_command --help`

或：

`some_command help`

甚至是命令下面的字参数，子命令的更详细的语法，可以用：

`some_command help sub_command`

往往可以输出对应的命令的帮助信息

举例：

# 嵌入式的uboot的语法

U-Boot: Quick reference - CompuLab Wiki

> help - print online help

-> help xxx

可以输出子命令的具体的详细的语法

xxx是uboot的子命令，比如ls，cp，md等等

# iOS中包管理器Carthage的语法

Carthage/Artifacts.md at master · Carthage/Carthage

Carthage/Carthage: A simple, decentralized dependency manager for Cocoa

```
licrifandeMacBook-Pro:QorosSales crifan$ carthage help
Available commands:

   archive           Archives built frameworks into a zip that Carthage can use
   bootstrap         Check out and build the project's dependencies
   build             Build the project's dependencies
   checkout          Check out the project's dependencies
   copy-frameworks   In a Run Script build phase, copies each framework specified by a SCRIPT_INPUT_FILE environment varia
ble into the built app bundle
   fetch             Clones or fetches a Git repository ahead of time
   help              Display general or command-specific help
   outdated          Check for compatible updates to the project's dependencies
   update            Update and rebuild the project's dependencies
   version           Display the current version of Carthage

licrifandeMacBook-Pro:QorosSales crifan$ carthage help build
Build the project's dependencies

[--configuration (string)]
the Xcode configuration to build

[--platform (platform)]
```

```
the platforms to build for (one of 'all', 'Mac', 'iOS', 'watchOS', 'tvOS', or comma-separated values of the formers except
for 'all')

[--derived-data (string)]
path to the custom derived data folder

[--no-skip-current]
don't skip building the Carthage project (in addition to its dependencies)

[--color (color)]
whether to apply color and terminal formatting (one of 'auto', 'always', or 'never')

[--verbose]
print xcodebuild output inline

[--project-directory (string)]
the directory containing the Carthage project

[[]]
the dependency names to build
```

# 苹果的Swift编译器 `swiftc`

之前在：

［已解决］Xcode项目编译出错：Command failed due to signal: Segmentation fault: 11

期间，注意到点苹果的swift语言的编译器swiftc，对应help结果是：

```
➜  crifanLib git:(master) swiftc --help
OVERVIEW: Swift compiler

USAGE: swiftc [options] inputs

MODES:
  -dump-ast               Parse and type-check input file(s) and dump AST(s)
  -dump-parse             Parse input file(s) and dump AST(s)
  -dump-scope-maps <expanded-or-list-of-line:column>
                          Parse and type-check input file(s) and dump the scope map(s)
  -dump-type-refinement-contexts
                          Type-check input file(s) and dump type refinement contexts(s)
  -emit-assembly          Emit assembly file(s) (-S)
  -emit-bc                Emit LLVM BC file(s)
...
  -parse                  Parse input file(s)
  -print-ast              Parse and type-check input file(s) and pretty print AST(s)
  -typecheck              Parse and type-check input file(s)

OPTIONS:
  -api-diff-data-file <path>
                           API migration data is from <path>
  -application-extension  Restrict code to those available for App Extensions
  -assert-config <value>  Specify the assert_configuration replacement. Possible values are Debug, Release, Unchecked, Dis
ableReplacement.
...
  -whole-module-optimization
                           Optimize input files together instead of individually
  -Xcc <arg>               Pass <arg> to the C/C++/Objective-C compiler
  -Xlinker <value>         Specifies an option which should be passed to the linker
```

# sftp命令不了解，用help去查看有哪些命令

之前：

【已解决】命令行中ftp操作上传到远程服务器crifan.com

期间，对于sftp的命令有哪些不是很了解，所以可以通过help去查看：

```
sftp> help
Available commands:
bye                                Quit sftp
cd path                            Change remote directory to 'path'
chgrp grp path                     Change group of file 'path' to 'grp'
chmod mode path                    Change permissions of file 'path' to 'mode'
chown own path                     Change owner of file 'path' to 'own'
df [-hi] [path]                    Display statistics for current directory or
                                   filesystem containing 'path'
exit                               Quit sftp
get [-afPpRr] remote [local]       Download file
reget [-fPpRr] remote [local]      Resume download file
reput [-fPpRr] [local] remote      Resume upload file
help                               Display this help text
lcd path                           Change local directory to 'path'
lls [ls-options [path]]            Display local directory listing
lmkdir path                        Create local directory
ln [-s] oldpath newpath            Link remote file (-s for symlink)
lpwd                               Print local working directory
ls [-1afhlnrSt] [path]             Display remote directory listing
lumask umask                       Set local umask to 'umask'
mkdir path                         Create remote directory
progress                           Toggle display of progress meter
put [-afPpRr] local [remote]       Upload file
pwd                                Display remote working directory
quit                               Quit sftp
rename oldpath newpath             Rename remote file
rm path                            Delete remote file
rmdir path                         Remove remote directory
symlink oldpath newpath            Symlink remote file
version                            Show SFTP version
!command                           Execute 'command' in local shell
!                                  Escape to local shell
?                                  Synonym for help
sftp>
```

# Next Generation

有些软件，代码，功能，在开发了新一代，下一代，功能更好更强的版本后，有的会以`Next Generation`去命令。

比如之前的：

## 嵌入式的交叉编译的工具`crosstool-ng`

crosstool 的下一代，新一代版本，叫做：crosstool-ng

## 用于科学上网的shadowsocks的Mac版本`Shadowsocks-NG`

之前叫做ShadowsocksX，其新版本，下一代，叫做Shadowsocks-NG

# 配置文件名为xxxfile

有很多技术、工具和框架，去给自己的配置文件命名的时候，用：`xxxfile[.yyy]`

比如：

## 前端开发

- `grunt`：`Gruntfile.js`
- `gulp`：`guplfile.js`

## iOS开发

- `Carthage`：`Cartfile`
- `Cocoapods`：`Podfile`

# i18n和l10n

在开发软件、APP时，往往会涉及到除了支持中文之外，还要支持其他国家的语言，比如英语、法语、德语等等，专业叫法叫做：**多国语言**（版本）或者**国际化**

而在谈论多国语言、国际化时，常常会遇到这个词：`i18n`

其实 `i18n` == `internationalization` == `国际化` == `多国语言`

-> 之所以这么叫是由于 `internationalization` 单词太长，把中间的**18**个字母 `internationalizatio` 省略掉，简写为了 `i18n`

和 `i18n` 的类似的含义的类似叫法还有：`l10n` == `localization` == `本地化`

举例：

之前见过的： python-babel/flask-babel: i18n and l10n support for Flask based on Babel and pytz

# 知识技能图谱

在学习不同领域的技术时，往往会发现，不同领域所需要学习的内容和方向差距还是很大的，感觉算是**隔行如隔山**，换个技术领域和方向，很多名词甚至都没听说过。

另外，在某个领域有了深入专研之后，往往可以总结出很多该领域相关的技术和名词，整理出成系统的内容，往往就叫做该领域的**技术图谱**，**知识图谱**。

# TeamStuQ/skill-map: 程序员技能图谱

这个TeamStuQ/skill-map: 程序员技能图谱中已经整理了几十个相关领域的技能图谱了，此处就不再做重复劳动，请移步去看即可。

此处把其他一些人制作的的图谱整理如下，供参考：

# 前端开发 知识图谱

QzhouZ/Front-End-Study: 前端开发基础知识学习


Web前端
HTML
HTML文档结构
段落标签
文字标签
图片标签
超链接标签
列表标签
表格标签
表单标签
框架标签
HTML5新增标签和属性
表单标签
canvas标签
媒体标签
CSS
CSS语法
CSS选择器
ID选择器
类选择器
复合选择器
CSS样式
背景
文本
字体
链接
列表
表格
盒子模型
边框
外边距
内边距
CSS定位
相对定位
绝对定位
浮动
CSS3新增属性
CSS布局实战
JavaScript
基本语法
变量
数据类型
函数
运算符
流程控制
JS对象
String对象
Array对象
Date对象
Window对象
Document对象
DOM操作
JS和CSS综合操作
jQuery
基础语法
选择器
基本选择器
层次选择器
过滤选择器
表单选择器
DOM操作
查找节点
创建节点
插入节点
删除节点
复制节点
替换节点
包裹节点
属性操作
样式操作
遍历节点
事件
事件绑定
合成事件
事件冒泡
移除事件
动画
show()、hide()
fadeIn()、fadeOut()
slideUp()、slideDown()
自定义动画animate()
动画回调函数
停止动画
表单、表格操作
常用jQuery插件
jQuery UI
综合实例
BootStrap
起步
栅格系统
排版
表格
表单
按钮
图片
模态框、弹出框、警告框
标签页
下拉菜单
按钮组、按钮式下拉菜单
导航、导航条
分页
标签、徽章
巨幕
BootStrap布局实例
响应式网站布局实战

By zkpplpp

[JacksonTian/fks: 前端技能汇总 Frontend Knowledge Structure]https://github.com/JacksonTian/fks

# Cheat Sheet 和 Handout

## Cheat sheet 的含义

软件领域内的

Cheat sheet和handout的解释

－》Cheat，中文含义为：作弊，Sheet：中文含义为：表格，表单，小纸片

－》cheat sheet，中文含义为：（用于作弊）小抄

－》放在软件行业内，意思是：

就像考试作弊一样，把大量的内容，精简的，有机的排列在一张纸，一张图上

特点：

- 化繁为简

用途：

- 方便随时参考和查阅
- 尤其常见于，把各种软件，工具等的快捷键，主要功能等，总结后列在一张图片内，多数以图文并茂的表格，图表，图片等形式展现

概述：

- 某某（语言，工具等）的Cheat sheet ＝＝ 某某（语言，工具等）快速查阅的图表型的手册

## handout 的含义

另外的 `handout`，中文本意：讲义；宣传册子，宣传品

－》往往有时候也可以认为：

`cheat sheet` == `handout`

Cheat sheet和handout的举例：

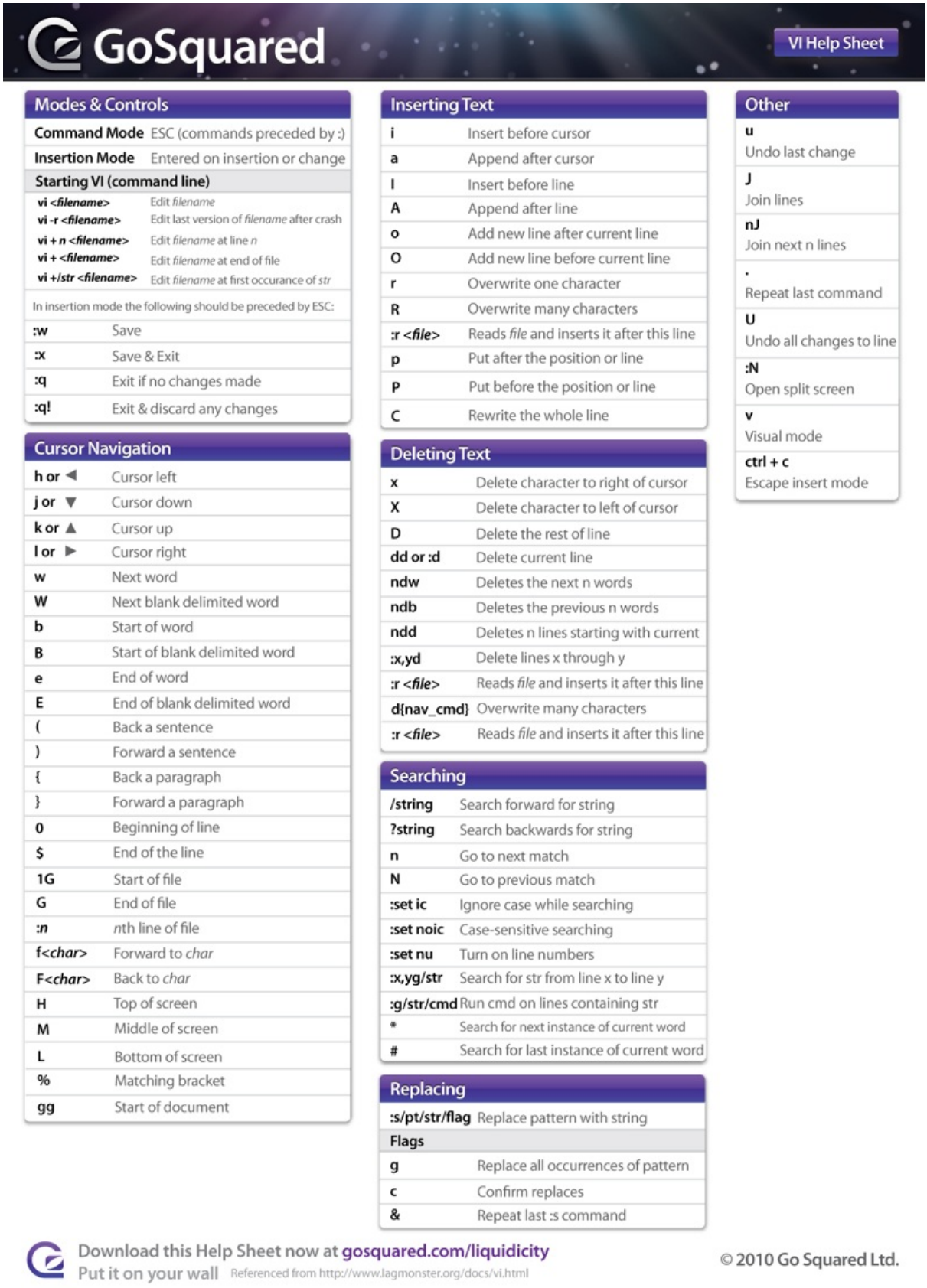
## Cheat sheet 举例

### vi的cheat sheet

VI Editor / Linux Terminal Cheat Sheet (PDF) – Smashing Magazine

－》

VI-Help-Sheet-01-large2.jpg



GoSquared

VI Help Sheet

**Modes & Controls**

| | |
|---|---|
| **Command Mode** | ESC (commands preceded by :) |
| **Insertion Mode** | Entered on insertion or change |

**Starting VI (command line)**

| | |
|---|---|
| **vi <filename>** | Edit *filename* |
| **vi -r <filename>** | Edit last version of *filename* after crash |
| **vi + n <filename>** | Edit *filename* at line *n* |
| **vi + <filename>** | Edit *filename* at end of file |
| **vi +/str <filename>** | Edit *filename* at first occurance of *str* |

In insertion mode the following should be preceded by ESC:

| | |
|---|---|
| **:w** | Save |
| **:x** | Save & Exit |
| **:q** | Exit if no changes made |
| **:q!** | Exit & discard any changes |

**Cursor Navigation**

| | |
|---|---|
| **h or ◄** | Cursor left |
| **j or ▼** | Cursor down |
| **k or ▲** | Cursor up |
| **l or ►** | Cursor right |
| **w** | Next word |
| **W** | Next blank delimited word |
| **b** | Start of word |
| **B** | Start of blank delimited word |
| **e** | End of word |
| **E** | End of blank delimited word |
| **(** | Back a sentence |
| **)** | Forward a sentence |
| **{** | Back a paragraph |
| **}** | Forward a paragraph |
| **0** | Beginning of line |
| **$** | End of the line |
| **1G** | Start of file |
| **G** | End of file |
| **:n** | *n*th line of file |
| **f<char>** | Forward to *char* |
| **F<char>** | Back to *char* |
| **H** | Top of screen |
| **M** | Middle of screen |
| **L** | Bottom of screen |
| **%** | Matching bracket |
| **gg** | Start of document |

**Inserting Text**

| | |
|---|---|
| **i** | Insert before cursor |
| **a** | Append after cursor |
| **I** | Insert before line |
| **A** | Append after line |
| **o** | Add new line after current line |
| **O** | Add new line before current line |
| **r** | Overwrite one character |
| **R** | Overwrite many characters |
| **:r <file>** | Reads *file* and inserts it after this line |
| **p** | Put after the position or line |
| **P** | Put before the position or line |
| **C** | Rewrite the whole line |

**Deleting Text**

| | |
|---|---|
| **x** | Delete character to right of cursor |
| **X** | Delete character to left of cursor |
| **D** | Delete the rest of line |
| **dd or :d** | Delete current line |
| **ndw** | Deletes the next n words |
| **ndb** | Deletes the previous n words |
| **ndd** | Deletes n lines starting with current |
| **:x,yd** | Delete lines x through y |
| **:r <file>** | Reads *file* and inserts it after this line |
| **d{nav_cmd}** | Overwrite many characters |
| **:r <file>** | Reads *file* and inserts it after this line |

**Searching**

| | |
|---|---|
| **/string** | Search forward for string |
| **?string** | Search backwards for string |
| **n** | Go to next match |
| **N** | Go to previous match |
| **:set ic** | Ignore case while searching |
| **:set noic** | Case-sensitive searching |
| **:set nu** | Turn on line numbers |
| **:x,yg/str** | Search for str from line x to line y |
| **:g/str/cmd** | Run cmd on lines containing str |
| ***** | Search for next instance of current word |
| **#** | Search for last instance of current word |

**Replacing**

| | |
|---|---|
| **:s/pt/str/flag** | Replace pattern with string |
| **Flags** | |
| **g** | Replace all occurrences of pattern |
| **c** | Confirm replaces |
| **&** | Repeat last :s command |

**Other**

| | |
|---|---|
| **u** | Undo last change |
| **J** | Join lines |
| **nJ** | Join next n lines |
| **.** | Repeat last command |
| **U** | Undo all changes to line |
| **:N** | Open split screen |
| **v** | Visual mode |
| **ctrl + c** | Escape insert mode |

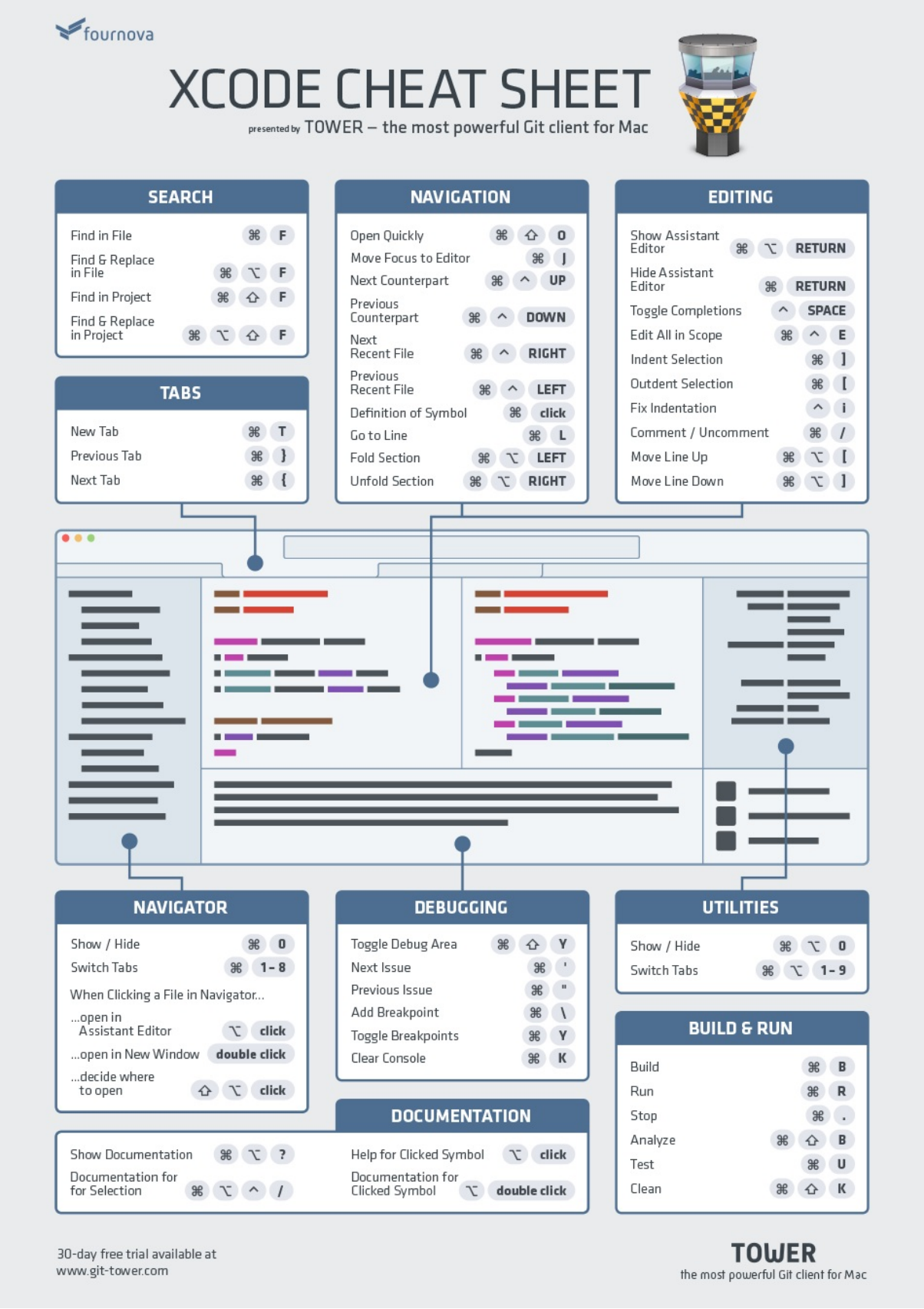
Download this Help Sheet now at gosquared.com/liquidicity
Put it on your wall Referenced from http://www.lagmonster.org/docs/vi.html



## Xcode的快捷键的cheat sheet

`http://www.crifan.com/xcode_common_used_keyboard_shortcut`

中的：



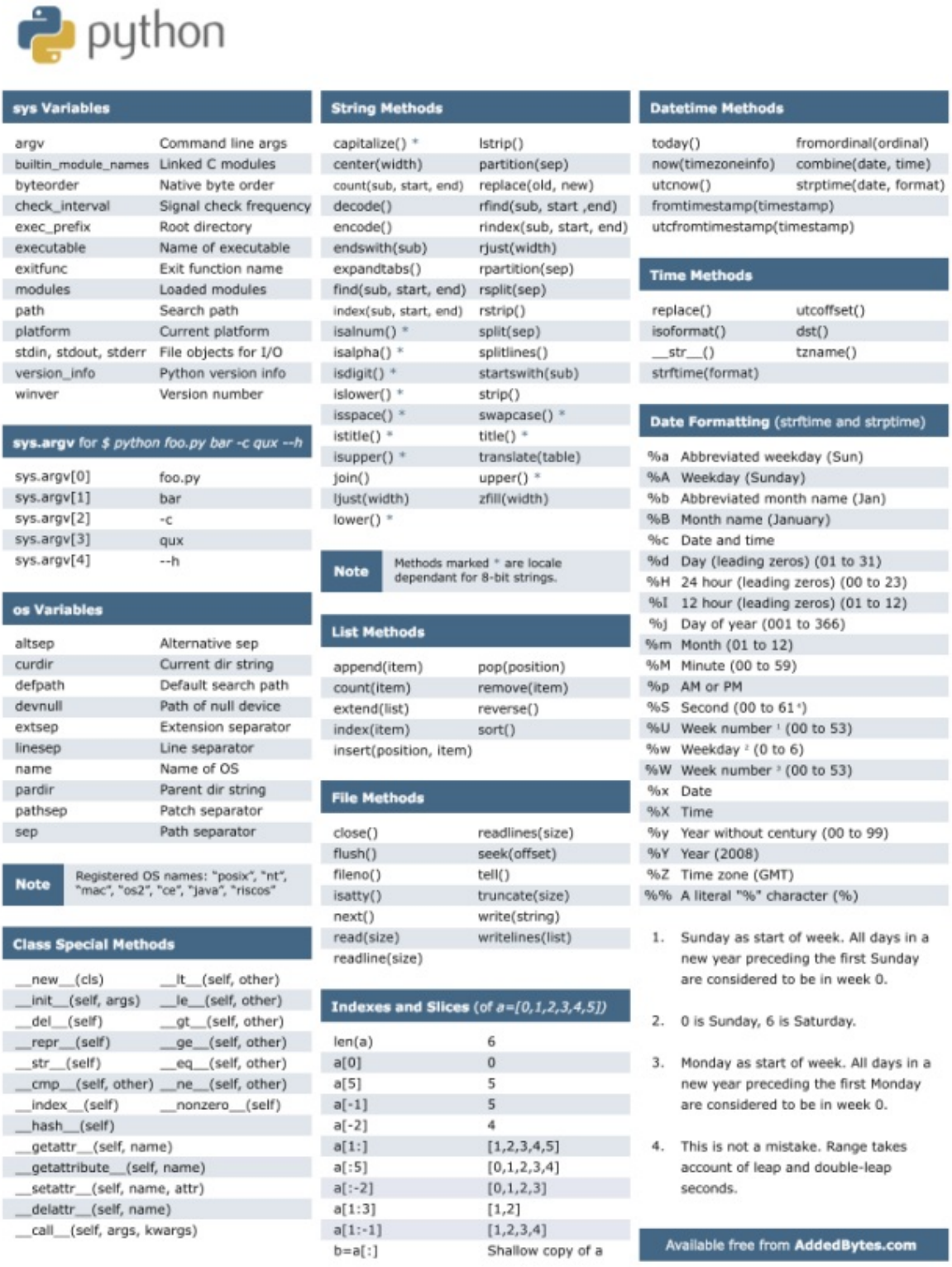

**python的cheat sheet**

python-cheat-sheet-v1.pdf

Python_qr.pdf

python:cours:mementopython3-english.pdf

Keep This Python Cheat Sheet on Hand When Learning to Code



python

**sys Variables**

| | |
|---|---|
| argv | Command line args |
| builtin_module_names | Linked C modules |
| byteorder | Native byte order |
| check_interval | Signal check frequency |
| exec_prefix | Root directory |
| executable | Name of executable |
| exitfunc | Exit function name |
| modules | Loaded modules |
| path | Search path |
| platform | Current platform |
| stdin, stdout, stderr | File objects for I/O |
| version_info | Python version info |
| winver | Version number |

**sys.argv** for *$ python foo.py bar -c qux --h*

| | |
|---|---|
| sys.argv[0] | foo.py |
| sys.argv[1] | bar |
| sys.argv[2] | -c |
| sys.argv[3] | qux |
| sys.argv[4] | --h |

**os Variables**

| | |
|---|---|
| altsep | Alternative sep |
| curdir | Current dir string |
| defpath | Default search path |
| devnull | Path of null device |
| extsep | Extension separator |
| linesep | Line separator |
| name | Name of OS |
| pardir | Parent dir string |
| pathsep | Patch separator |
| sep | Path separator |

**Note** Registered OS names: "posix", "nt", "mac", "os2", "ce", "java", "riscos"

**Class Special Methods**

| | |
|---|---|
| __new__(cls) | __lt__(self, other) |
| __init__(self, args) | __le__(self, other) |
| __del__(self) | __gt__(self, other) |
| __repr__(self) | __ge__(self, other) |
| __str__(self) | __eq__(self, other) |
| __cmp__(self, other) | __ne__(self, other) |
| __index__(self) | __nonzero__(self) |
| __hash__(self) | |
| __getattr__(self, name) | |
| __getattribute__(self, name) | |
| __setattr__(self, name, attr) | |
| __delattr__(self, name) | |
| __call__(self, args, kwargs) | |

**String Methods**

| | |
|---|---|
| capitalize() * | lstrip() |
| center(width) | partition(sep) |
| count(sub, start, end) | replace(old, new) |
| decode() | rfind(sub, start ,end) |
| encode() | rindex(sub, start, end) |
| endswith(sub) | rjust(width) |
| expandtabs() | rpartition(sep) |
| find(sub, start, end) | rsplit(sep) |
| index(sub, start, end) | rstrip() |
| isalnum() * | split(sep) |
| isalpha() * | splitlines() |
| isdigit() * | startswith(sub) |
| islower() * | strip() |
| isspace() * | swapcase() * |
| istitle() * | title() * |
| isupper() * | translate(table) |
| join() | upper() * |
| ljust(width) | zfill(width) |
| lower() * | |

**Note** Methods marked * are locale dependant for 8-bit strings.

**List Methods**

| | |
|---|---|
| append(item) | pop(position) |
| count(item) | remove(item) |
| extend(list) | reverse() |
| index(item) | sort() |
| insert(position, item) | |

**File Methods**

| | |
|---|---|
| close() | readlines(size) |
| flush() | seek(offset) |
| fileno() | tell() |
| isatty() | truncate(size) |
| next() | write(string) |
| read(size) | writelines(list) |
| readline(size) | |

**Indexes and Slices** (of *a=[0,1,2,3,4,5]*)

| | |
|---|---|
| len(a) | 6 |
| a[0] | 0 |
| a[5] | 5 |
| a[-1] | 5 |
| a[-2] | 4 |
| a[1:] | [1,2,3,4,5] |
| a[:5] | [0,1,2,3,4] |
| a[:-2] | [0,1,2,3] |
| a[1:3] | [1,2] |
| a[1:-1] | [1,2,3,4] |
| b=a[:] | Shallow copy of a |

**Datetime Methods**

| | |
|---|---|
| today() | fromordinal(ordinal) |
| now(timezoneinfo) | combine(date, time) |
| utcnow() | strptime(date, format) |
| fromtimestamp(timestamp) | |
| utcfromtimestamp(timestamp) | |

**Time Methods**

| | |
|---|---|
| replace() | utcoffset() |
| isoformat() | dst() |
| __str__() | tzname() |
| strftime(format) | |

**Date Formatting** (strftime and strptime)

| | |
|---|---|
| %a | Abbreviated weekday (Sun) |
| %A | Weekday (Sunday) |
| %b | Abbreviated month name (Jan) |
| %B | Month name (January) |
| %c | Date and time |
| %d | Day (leading zeros) (01 to 31) |
| %H | 24 hour (leading zeros) (00 to 23) |
| %I | 12 hour (leading zeros) (01 to 12) |
| %j | Day of year (001 to 366) |
| %m | Month (01 to 12) |
| %M | Minute (00 to 59) |
| %p | AM or PM |
| %S | Second (00 to 61 [4]) |
| %U | Week number [1] (00 to 53) |
| %w | Weekday [2] (0 to 6) |
| %W | Week number [3] (00 to 53) |
| %x | Date |
| %X | Time |
| %y | Year without century (00 to 99) |
| %Y | Year (2008) |
| %Z | Time zone (GMT) |
| %% | A literal "%" character (%) |

1. Sunday as start of week. All days in a new year preceding the first Sunday are considered to be in week 0.
2. 0 is Sunday, 6 is Saturday.
3. Monday as start of week. All days in a new year preceding the first Monday are considered to be in week 0.
4. This is not a mistake. Range takes account of leap and double-leap seconds.

Available free from **AddedBytes.com**

## go语言的cheat sheet

go-lang-cheat-sheet

# Go Cheat Sheet

**Credits**

Most example code taken from A Tour of Go, which is an excellent introduction to Go. If you're new to Go, do that tour. Seriously.

Original HTML Cheat Sheet by Ariel Mashraki (a8m):
https://github.com/a8m/go-lang-cheat-sheet

## Go in a Nutshell

- Imperative language
- Statically typed
- Syntax similar to Java/C/C++, but less parentheses and no semicolons
- Compiles to native code (no JVM)
- No classes, but structs with methods
- Interfaces
- No implementation inheritance. There's type embedding, though.
- Functions are first class citizens
- Functions can return multiple values
- Has closures
- Pointers, but not pointer arithmetic
- Built-in concurrency primitives: Goroutines and Channels

## Basic Syntax

### Hello World

File `hello.go`:

```
package main

import "fmt"

func main() {
    fmt.Println("Hello Go")
}
```

```
$ go run hello.go
```

## Operators

### Arithmetic

| Operator | Description |
|---|---|
| + | addition |
| - | subtraction |
| * | multiplication |
| / | quotient |
| % | remainder |
| & | bitwise AND |
| \| | bitwise OR |
| ^ | bitwise XOR |
| &^ | bit clear (AND NOT) |
| << | left shift |
| >> | right shift (logical) |

### Comparison

| Operator | Description |
|---|---|
| == | equal |
| != | not equal |
| < | less than |
| <= | less than or equal |
| > | greater than |
| >= | greater than or equal |

### Logical

| Operator | Description |
|---|---|
| && | logical AND |
| \|\| | logical OR |
| ! | logical NOT |

### Other

| Operator | Description |
|---|---|
| & | address of / create pointer |
| * | dereference pointer |
| <- | send / receive operator (see 'Channels' below) |

## Declarations

Type goes **after** identifier!

```
var foo int                 // declaration without initialization
var foo int = 42            // declaration with initialization
var foo, bar int = 42, 1302  // declare/init multiple vars at once
var foo = 42                // type omitted, will be inferred
foo := 42  // shorthand, only in func bodies, implicit type
const constant = "This is a constant"
```

## Functions

```
// a simple function
func functionName() {}

// function with parameters (again, types go after identifiers)
func functionName(param1 string, param2 int) {}

// multiple parameters of the same type
func functionName(param1, param2 int) {}
```

## Functions (cont)

```
// return type declaration
func functionName() int {
    return 42
}

// Can return multiple values at once
func returnMulti() (int, string) {
    return 42, "foobar"
}
var x, str = returnMulti()

// Return multiple named results simply by return
func returnMulti2() (n int, s string) {
    n = 42
    s = "foobar"
    // n and s will be returned
    return
}
var x, str = returnMulti2()
```

### Functions As Values And Closures

```
func main() {
    // assign a function to a name
    add := func(a, b int) int {
        return a + b
    }
    // use the name to call the function
    fmt.Println(add(3, 4))
}

// Closures, lexically scoped: Functions can access values that were
// in scope when defining the function
func scope() func() int{
    outer_var := 2
    foo := func() int { return outer_var}
    return foo
}

func another_scope() func() int{
    // won't compile - outer_var and foo not defined in this scope
    outer_var = 444
    return foo
}

// Closures: don't mutate outer vars, instead redefine them!
func outer() (func() int, int) {
    outer_var := 2       // NOTE outer_var is outside inner's scope
    inner := func() int {
        outer_var += 99  // attempt to mutate outer_var
        return outer_var // => 101 (but outer_var is a newly redefined
                         //  variable visible only inside inner)
    }
    return inner, outer_var // => 101, 2 (still!)
}
```

## Functions (cont)

### Variadic Functions

```
func main() {
    fmt.Println(adder(1, 2, 3)) // 6
    fmt.Println(adder(9, 9))    // 18

    nums := []int{10, 20, 30}
    fmt.Println(adder(nums...)) // 60
}

// Using  ... before the type name of the last parameter indicates
//  that it takes zero or more of those parameters.
// The function is invoked like any other function except we can
//  pass as many arguments as we want.
func adder(args ...int) int {
    total := 0
    for _, v := range args { // Iterate over all args
        total += v
    }
    return total
}
```

## Built-in Types

```
bool

string

int  int8  int16  int32  int64
uint uint8 uint16 uint32 uint64 uintptr

byte // alias for uint8

rune // alias for int32 ~= (Unicode code point) - Very Viking

float32 float64

complex64 complex128
```

## Type Conversions

```
var i int = 42
var f float64 = float64(i)
var u uint = uint(f)

// alternative syntax
i := 42
f := float64(i)
u := uint(f)
```

## Packages

## Control structures (cont)

- package declaration at top of every source file
- executables are in package `main`
- convention: package name == last name of import path (import path `math/rand` => package `rand`)
- upper case identifier: exported (visible from other packages)
- Lower case identifier: private (not visible from other packages)

## Control structures

### If

```
func main() {
    // Basic one
    if x > 0 {
        return x
    } else {
        return -x
    }

    // You can put one statement before the condition
    if a := b + c; a < 42 {
        return a
    } else {
        return a - 42
    }

    // Type assertion inside if
    var val interface{}
    val = "foo"
    if str, ok := val.(string); ok {
        fmt.Println(str)
    }
}
```

### Loops

```
    // There's only `for`. No `while`, no `until`
    for i := 1; i < 10; i++ {
    }
    for ; i < 10;  { // while loop
    }
    for i < 10  { // can omit semicolons if there's only a condition
    }
    for {         // can omit the condition ~ while (true)
    }
```

### Switch

```
    // switch statement
    switch operatingSystem {
    case "darwin":
        fmt.Println("Mac OS Hipster")
        // cases break automatically, no fallthrough by default
    case "linux":
        fmt.Println("Linux Geek")
    default:
        // Windows, BSD, ...
        fmt.Println("Other")
    }

    // As with for and if, an assignment statement before the
    //  switch value is allowed
    switch os := runtime.GOOS; os {
        case "darwin": ...
    }
```

## Arrays, Slices, Ranges

### Arrays

```
var a [10]int  // int array with length 10. Length is part of type!
a[3] = 42      // set elements
i := a[3]      // read elements

// declare and initialize
var a = [2]int{1, 2}
a := [2]int{1, 2}    //shorthand
a := [...]int{1, 2}  // elipsis -> Compiler figures out array length
```

### Slices

```
var a []int    // a slice – like an array, but length is unspecified
var a = []int {1, 2, 3, 4}  // declare and initialize a slice
                            //  (backed by given array implicitly)
a := []int{1, 2, 3, 4}      // shorthand
chars := []string{0:"a", 2:"c", 1:"b"}  // ["a", "b", "c"]

var b = a[lo:hi]    // creates a slice (view of the array) from
                    //  index lo to hi-1
var b = a[1:4]      // slice from index 1 to 3
var b = a[:3]       // missing low index implies 0
var b = a[3:]       // missing high index implies len(a)

// create a slice with make
a = make([]byte, 5, 5)  // first arg length, second capacity
a = make([]byte, 5)     // capacity is optional
```

## Arrays, Slices, Ranges (cont)

```
// create a slice from an array
x := [3]string{"Лайка", "Белка", "Стрелка"}
s := x[:]  // a slice referencing the storage of x
```

### Operations on Arrays and Slices

`len(a)` gives you the length of an array/a slice. It's a built-in function, not a attribute/method on the array.

```
// loop over an array/a slice
for i, e := range a {
    // i is the index, e the element
}

// if you only need e:
for _, e := range a {
    // e is the element
}

// ...and if you only need the index
for i := range a {
}

// In Go pre-1.4, it is a compiler error to not use i and e.
// Go 1.4 introduced a variable-free form:
for range time.Tick(time.Second) {
    // do it once a sec
}
```

## Maps

```
var m map[string]int
m = make(map[string]int)
m["key"] = 42
fmt.Println(m["key"])

delete(m, "key")

elem, ok := m["key"] // test if key "key" is present, retrieve if so

// map literal
var m = map[string]Vertex{
    "Bell Labs": {40.68433, -74.39967},
    "Google":    {37.42202, -122.08408},
}
```

## Structs

There are no classes, only structs. Structs can have methods.

```
// A struct is a type. It's also a collection of fields

// Declaration
type Vertex struct {
    X, Y int
}

// Creating
var v = Vertex{1, 2}
var v = Vertex{X: 1, Y: 2} // Creates a struct by defining values
                           //  with keys

// Accessing members
v.X = 4

// You can declare methods on structs. The struct you want to declare
//  the method on (the receiving type) comes between the func keyword
//  and the method name. The struct is copied on each method call(!)
func (v Vertex) Abs() float64 {
    return math.Sqrt(v.X*v.X + v.Y*v.Y)
}

// Call method
v.Abs()

// For mutating methods, you need to use a pointer (see below) to the
//  Struct as the type. With this, the struct value is not copied for
//  the method call.
func (v *Vertex) add(n float64) {
    v.X += n
    v.Y += n
}
```

## Anonymous structs

Cheaper and safer than using `map[string]interface{}`.

```
point := struct {
    X, Y int
}{1, 2}
```

## Pointers

```
p := Vertex{1, 2}  // p is a Vertex
q := &p            // q is a pointer to a Vertex
r := &Vertex{1, 2} // r is also a pointer to a Vertex

// The type of a pointer to a Vertex is *Vertex

var s *Vertex = new(Vertex) // create ptr to a new struct instance
```

## Interfaces

```
// interface declaration
type Awesomizer interface {
    Awesomize() string
}

// types do *not* declare to implement interfaces
type Foo struct {}

// instead, types implicitly satisfy an interface if they implement
all required methods
func (foo Foo) Awesomize() string {
    return "Awesome!"
}
```

## Embedding

There is no subclassing in Go. Instead, there is interface and struct embedding (composition).

```
// ReadWriter implementations must satisfy both Reader and Writer
type ReadWriter interface {
    Reader
    Writer
}

// Server exposes all the methods that Logger has
type Server struct {
    Host string
    Port int
    *log.Logger
}

// initialize the embedded type the usual way
server := &Server{"localhost", 80, log.New(...)}

// methods implemented on the embedded struct are passed through
server.Log(...) // calls server.Logger.Log(...)

// Field name of an embedded type is its type name ('Logger' here)
var logger *log.Logger = server.Logger
```

## Errors

There is no exception handling. Functions that might produce an error just declare an additional return value of type `Error`. This is the `Error` interface:

```
type error interface {
    Error() string
}
```

## Errors (cont)

A function that might return an error:

```
func doStuff() (int, error) {
}

func main() {
    result, error := doStuff()
    if (error != nil) {
        // handle error
    } else {
        // all is good, use result
    }
}
```

## Concurrency

## Goroutines

Goroutines are lightweight threads (managed by Go, not OS threads). `go f(a, b)` starts a new goroutine which runs `f` (given `f` is a function).

```
// just a function (which can be later started as a goroutine)
func doStuff(s string) {
}

func main() {
    // using a named function in a goroutine
    go doStuff("foobar")

    // using an anonymous inner function in a goroutine
    go func (x int) {
        // function body goes here
    }(42)
}
```

## Channels

```
ch := make(chan int) // create a channel of type int
ch <- 42             // Send a value to the channel ch.
v := <-ch            // Receive a value from ch

// Non-buffered channels block. Read blocks when no value is
// available, write blocks if a value already has been written
// but not read.

// Create a buffered channel. Writing to a buffered channels does
// not block if less than <buffer size> unread values have been
// written.
ch := make(chan int, 100)

close(c) // closes the channel (only sender should close)

// Read from channel and test if it has been closed
// If ok is false, channel has been closed
v, ok := <-ch

// Read from channel until it is closed
for i := range ch {
    fmt.Println(i)
}

// select blocks on multiple channel operations.
// If one unblocks, the corresponding case is executed
func doStuff(channelOut, channelIn chan int) {
    select {
    case channelOut <- 42:
        fmt.Println("We could write to channelOut!")
    case x := <- channelIn:
        fmt.Println("We could read from channelIn")
    case <-time.After(time.Second * 1):
        fmt.Println("timeout")
    }
}
```

Channel Axioms

```
// I. A send to a nil channel blocks forever
var c chan string
c <- "Hello, World!"
// fatal error: all goroutines are asleep - deadlock!

// II. A receive from a nil channel blocks forever
var c chan string
fmt.Println(<-c)
// fatal error: all goroutines are asleep - deadlock!

// III. A send to a closed channel panics
var c = make(chan string, 1)
c <- "Hello, World!"
close(c)
```

## Channels (cont)

```
c <- "Hello, Panic!"
// panic: send on closed channel

// IV. A receive from a close channel returns the zero value
// immediately
var c = make(chan int, 2)
c <- 1
c <- 2
close(c)
for i := 0; i < 3; i++ {
    fmt.Printf("%d ", <-c)
}
// 1 2 0
```

## Snippets

## HTTP Server

```
package main

import (
    "fmt"
    "net/http"
)

// define a type for the response
type Hello struct{}

// let that type implement the ServeHTTP method (defined in
// interface http.Handler)
func (h Hello) ServeHTTP(w http.ResponseWriter, r *http.Request) {
    fmt.Fprint(w, "Hello!")
}

func main() {
    var h Hello
    http.ListenAndServe("localhost:4000", h)
}

// Here's the method signature of http.ServeHTTP:
// type Handler interface {
//     ServeHTTP(w http.ResponseWriter, r *http.Request)
// }
```

## HTML的cheat sheet

htmlcheatsheet.pdf

HTML Cheat Sheet - A Simple Guide to HTML

HTML5 Cheat Sheet - WebsiteSetup.org

HTML5 Cheat Sheet - WebsiteSetup.org

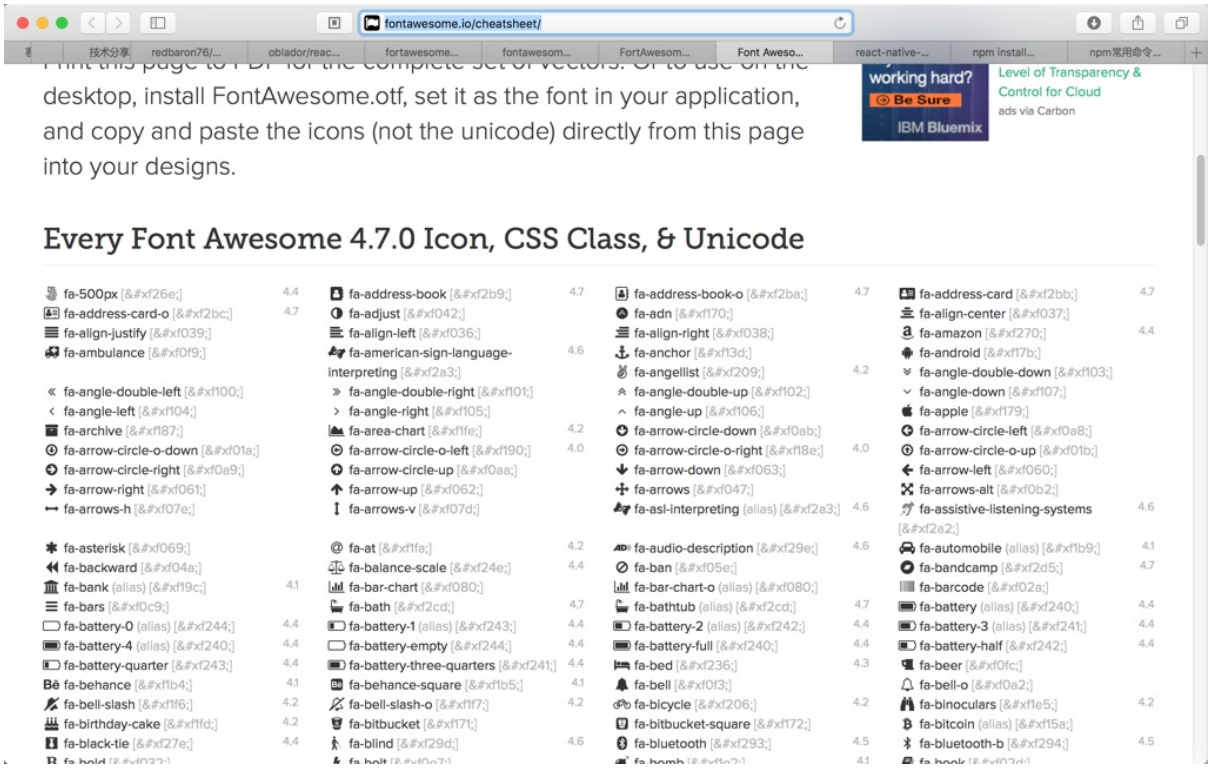

## Font Awesome 的Cheatsheet

React Native折腾期间，需要利用到对应字体：fortawesome.github.io/Font-Awesome/icons/

-》

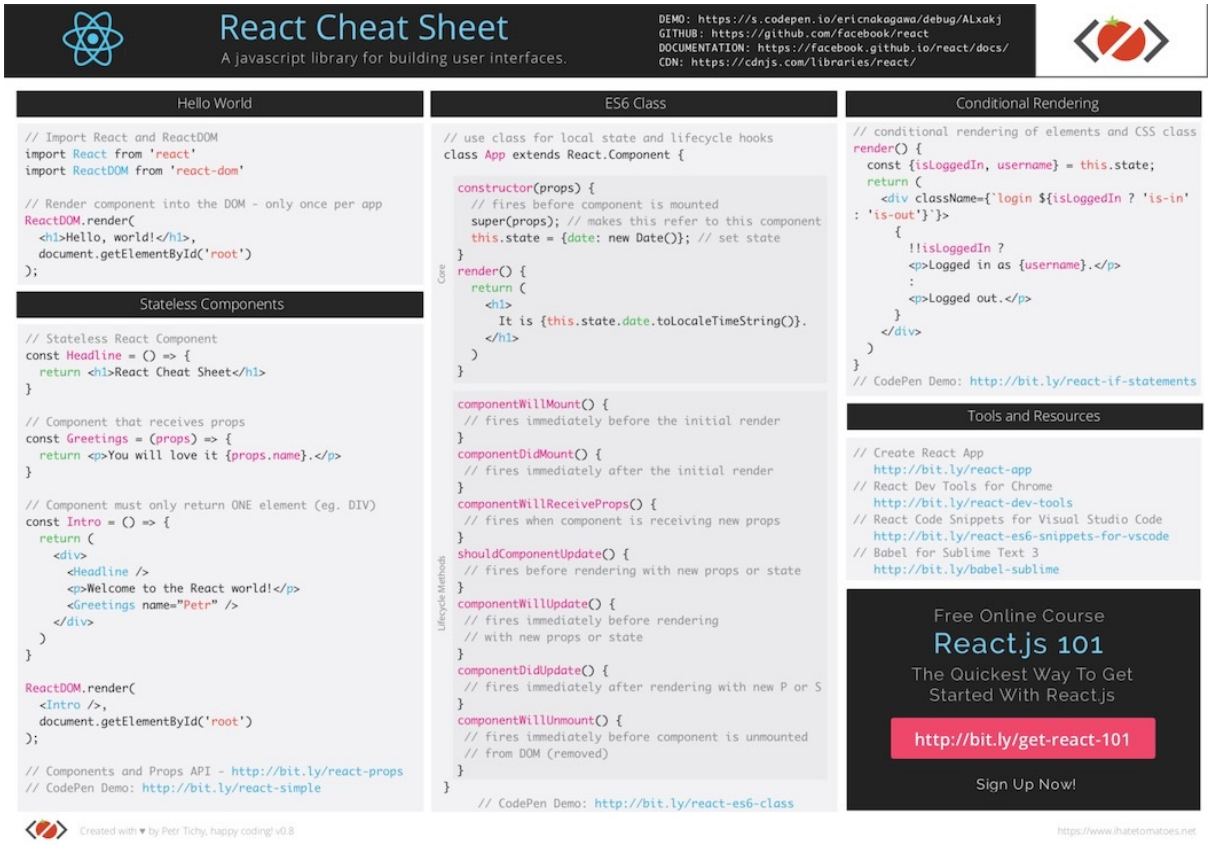
Font Awesome Cheatsheet

就列出了所有的字体图标



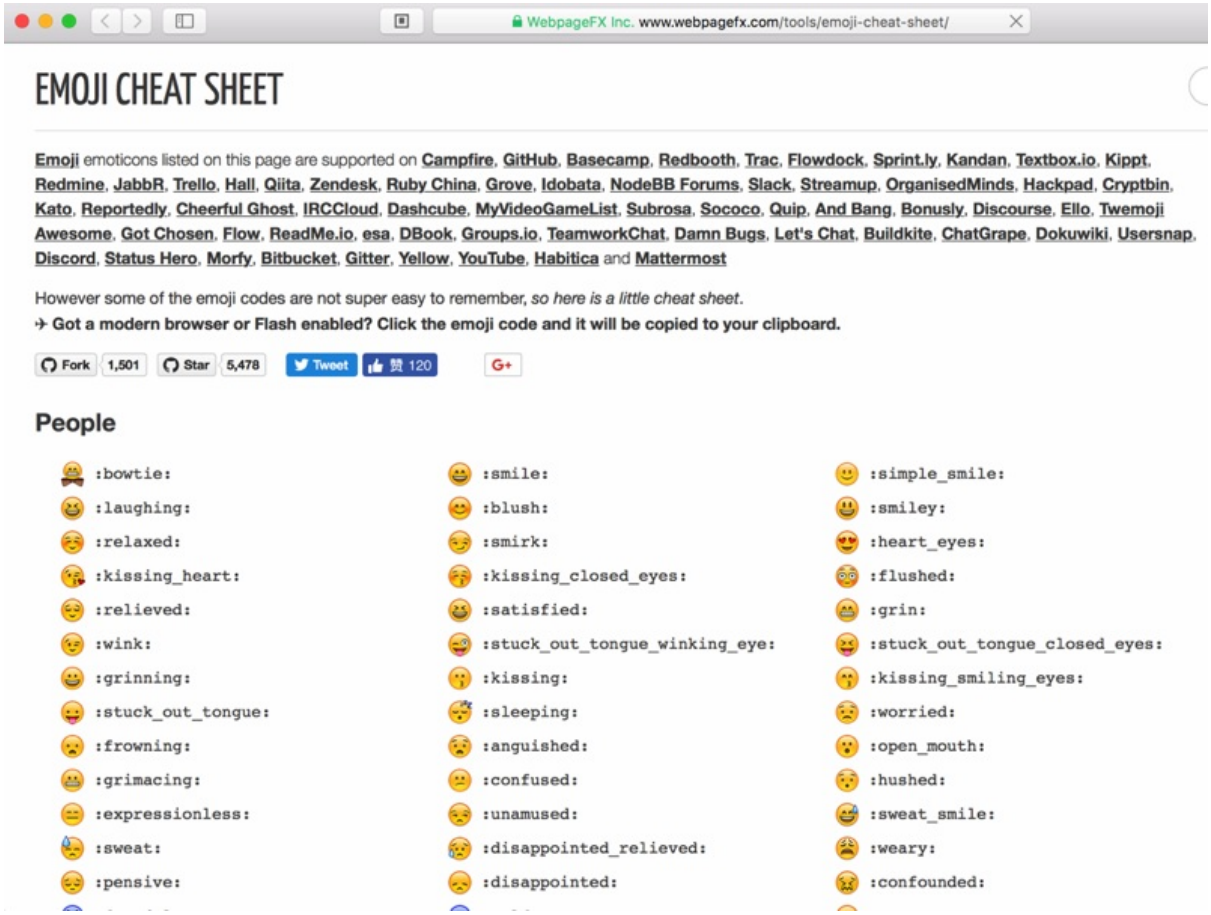

## React 的cheat sheet

React Cheat Sheet




## Emoji的cheat sheet

Emoji cheat sheet for GitHub, Basecamp and other services

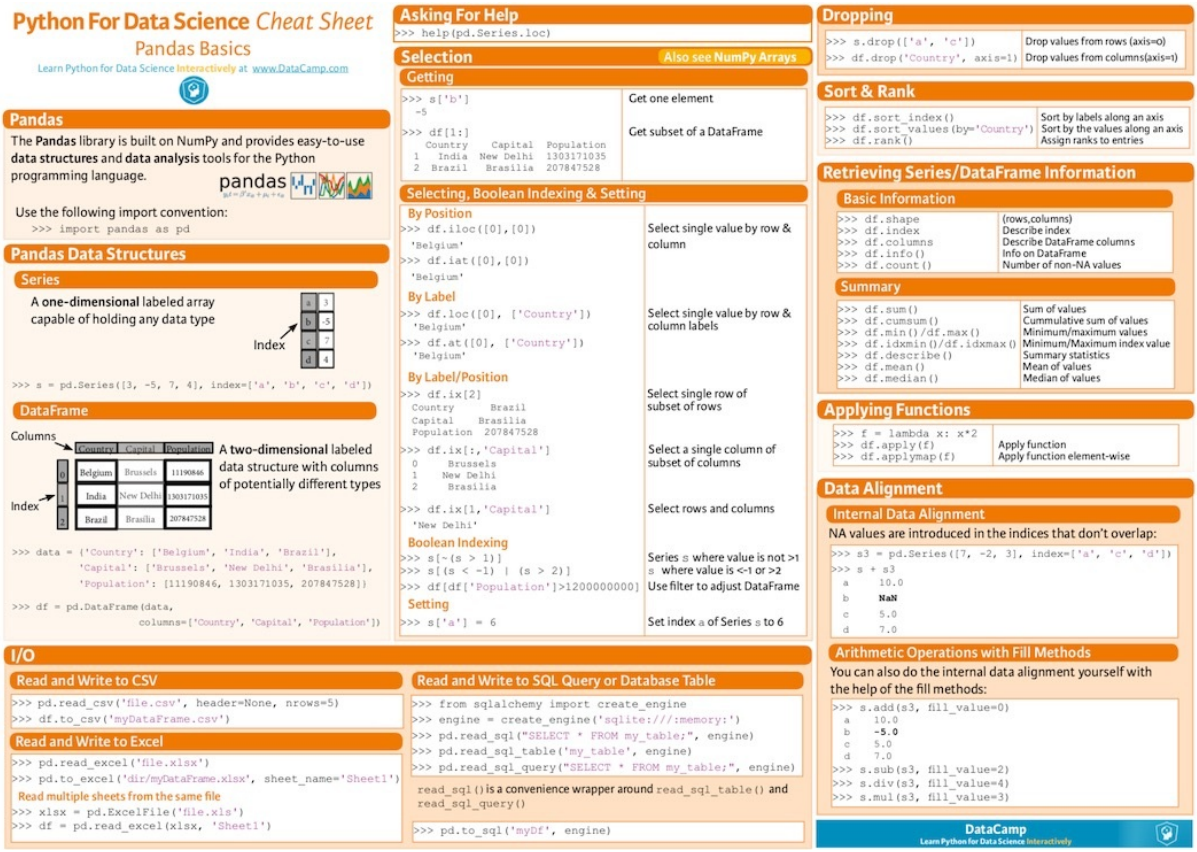



## Python的Pandas的cheatsheet

Pandas Cheat Sheet for Data Science in Python (article) - DataCamp

-》

PandasPythonForDataScience.pdf

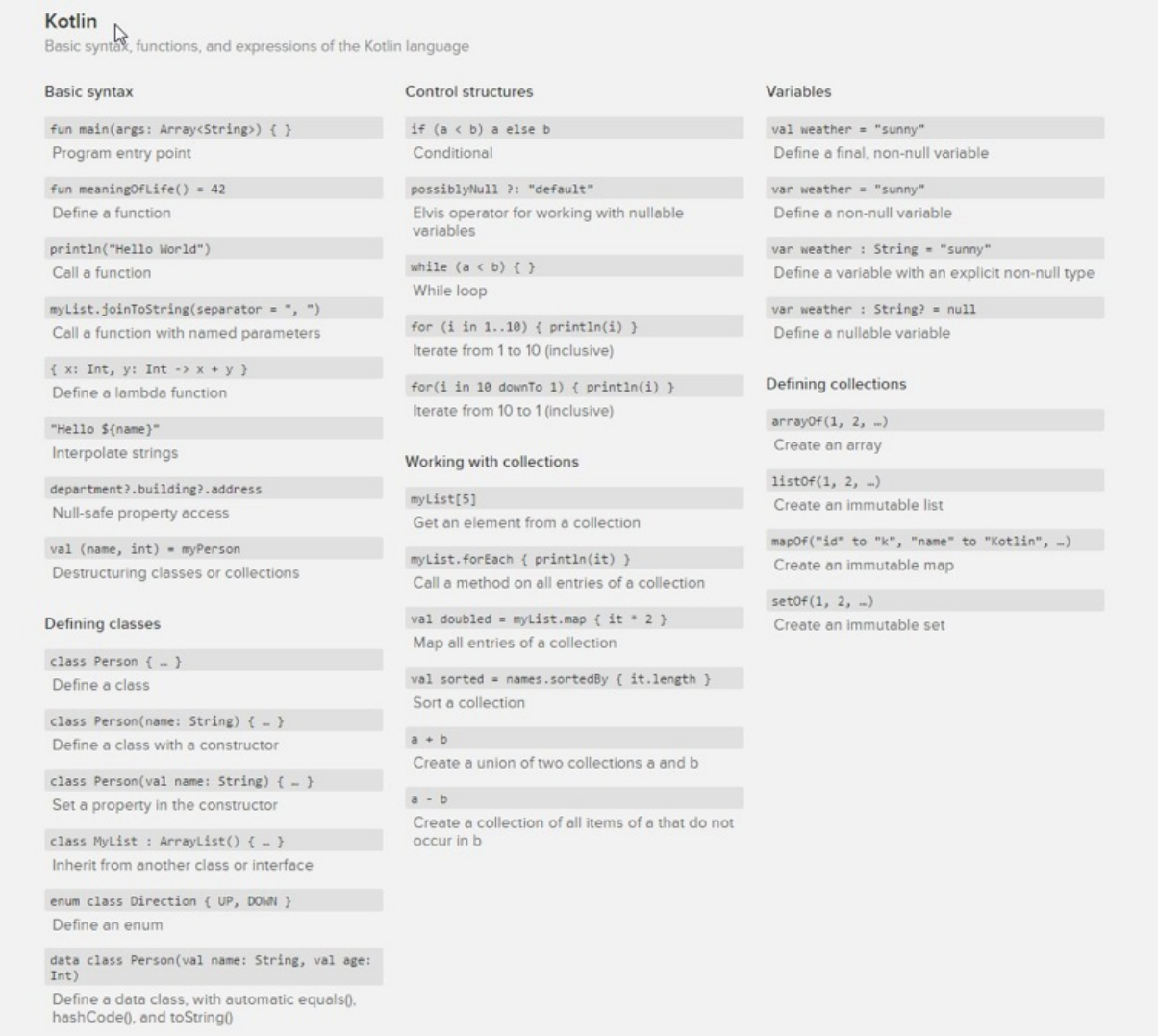



## Kotlin的cheatsheet

Kotlin 作为 Android 开发语言相比传统 Java 有什么优势？ - 知乎

Kotlin Programming Language Cheat Sheet Part 1

Kotlin Programming Language Cheat Sheet Part 2



## Kotlin

Basic syntax, functions, and expressions of the Kotlin language

### Basic syntax

```
fun main(args: Array<String>) { }
```
Program entry point

```
fun meaningOfLife() = 42
```
Define a function

```
println("Hello World")
```
Call a function

```
myList.joinToString(separator = ", ")
```
Call a function with named parameters

```
{ x: Int, y: Int -> x + y }
```
Define a lambda function

```
"Hello ${name}"
```
Interpolate strings

```
department?.building?.address
```
Null-safe property access

```
val (name, int) = myPerson
```
Destructuring classes or collections

### Defining classes

```
class Person { … }
```
Define a class

```
class Person(name: String) { … }
```
Define a class with a constructor

```
class Person(val name: String) { … }
```
Set a property in the constructor

```
class MyList : ArrayList() { … }
```
Inherit from another class or interface

```
enum class Direction { UP, DOWN }
```
Define an enum

```
data class Person(val name: String, val age: Int)
```
Define a data class, with automatic equals(), hashCode(), and toString()

### Control structures

```
if (a < b) a else b
```
Conditional

```
possiblyNull ?: "default"
```
Elvis operator for working with nullable variables

```
while (a < b) { }
```
While loop

```
for (i in 1..10) { println(i) }
```
Iterate from 1 to 10 (inclusive)

```
for(i in 10 downTo 1) { println(i) }
```
Iterate from 10 to 1 (inclusive)

### Working with collections

```
myList[5]
```
Get an element from a collection

```
myList.forEach { println(it) }
```
Call a method on all entries of a collection

```
val doubled = myList.map { it * 2 }
```
Map all entries of a collection

```
val sorted = names.sortedBy { it.length }
```
Sort a collection

```
a + b
```
Create a union of two collections a and b

```
a - b
```
Create a collection of all items of a that do not occur in b

### Variables

```
val weather = "sunny"
```
Define a final, non-null variable

```
var weather = "sunny"
```
Define a non-null variable

```
var weather : String = "sunny"
```
Define a variable with an explicit non-null type

```
var weather : String? = null
```
Define a nullable variable

### Defining collections

```
arrayOf(1, 2, …)
```
Create an array

```
listOf(1, 2, …)
```
Create an immutable list

```
mapOf("id" to "k", "name" to "Kotlin", …)
```
Create an immutable map

```
setOf(1, 2, …)
```
Create an immutable set

# 编译和链接

此处介绍编译和链接方面的通用的知识、概念和逻辑。

之前接触过嵌入式和Linux，断断续续接触过很多编译和链接的东西，主要是命令行界面中用 `makefile` 、 `gcc` 等各种工具去做编译和链接的工作。

后来又接触过移动端开发，包括 `Xcode` 等IDE集成开发环境

> **[info] 编辑器和IDE**
>
> 详见：【整理】文本编辑器和IDE集成开发环境

发现其实IDE背后都是利用了编译和链接方面的通用知识。

对此编译和链接方面的通用的知识、逻辑、概念，去尝试总结，以供参考。

# Xcode中底层是用swiftc去编译swift代码

之前在：

［已解决］Xcode项目编译出错：Command failed due to signal: Segmentation fault: 11

期间，通过错误的log中的：

```
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/swiftc -incremental -module-name Sa
lesApp -Onone -sdk /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimu
lator9.3.sdk -target i386-apple-ios9.0 -g -module-cache-path /Users/crifan/Library/Developer/Xcode/DerivedData/ModuleCache
 -Xfrontend -serialize-debugging-options -enable-testing -I /Users/crifan/Library/Developer/Xcode/DerivedData/SalesApp-fxz
zwvtqreanqsgzqmztnatyyjjd/Build/Products/Debug-iphonesimulator -F /Users/crifan/Library/Developer/Xcode/DerivedData/SalesA
pp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Products/Debug-iphonesimulator -F /Users/crifan/dev/dev_root/daryun/Projects/Qoros/Q
orosSales/Sourcecode/SalesAppiOS/Carthage/Build/iOS -c -j4 /Users/crifan/dev/dev_root/daryun/Projects/Qoros/QorosSales/Sou
rcecode/SalesAppiOS/SalesApp/CrifanLibHttp.swift
...
/Users/crifan/dev/dev_root/daryun/Projects/Qoros/QorosSales/Sourcecode/SalesAppiOS/SalesApp/CrifanLibDemo.swift /Users/cri
fan/dev/dev_root/daryun/Projects/Qoros/QorosSales/Sourcecode/SalesAppiOS/SalesApp/AddTaskViewController.swift -output-file
-map /Users/crifan/Library/Developer/Xcode/DerivedData/SalesApp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Intermediates/SalesApp.
build/Debug-iphonesimulator/SalesApp.build/Objects-normal/i386/SalesApp-OutputFileMap.json -parseable-output -serialize-di
agnostics -emit-dependencies -emit-module -emit-module-path /Users/crifan/Library/Developer/Xcode/DerivedData/SalesApp-fxz
zwvtqreanqsgzqmztnatyyjjd/Build/Intermediates/SalesApp.build/Debug-iphonesimulator/SalesApp.build/Objects-normal/i386/Sale
sApp.swiftmodule -Xcc -I/Users/crifan/Library/Developer/Xcode/DerivedData/SalesApp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Inte
rmediates/SalesApp.build/Debug-iphonesimulator/SalesApp.build/swift-overrides.hmap -Xcc -iquote -Xcc /Users/crifan/Library
/Developer/Xcode/DerivedData/SalesApp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Intermediates/SalesApp.build/Debug-iphonesimulato
r/SalesApp.build/SalesApp-generated-files.hmap -Xcc -I/Users/crifan/Library/Developer/Xcode/DerivedData/SalesApp-fxzzwvtqr
eanqsgzqmztnatyyjjd/Build/Intermediates/SalesApp.build/Debug-iphonesimulator/SalesApp.build/SalesApp-own-target-headers.hm
ap
...
-Xcc -I/Users/crifan/Library/Developer/Xcode/DerivedData/SalesApp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Products/Debug-iphone
simulator/include -Xcc -I/Users/crifan/Library/Developer/Xcode/DerivedData/SalesApp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Int
ermediates/SalesApp.build/Debug-iphonesimulator/SalesApp.build/DerivedSources/i386 -Xcc -I/Users/crifan/Library/Developer/
Xcode/DerivedData/SalesApp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Intermediates/SalesApp.build/Debug-iphonesimulator/SalesApp.
build/DerivedSources -Xcc -DDEBUG 1 -emit-objc-header -emit-objc-header-path /Users/crifan/Library/Developer/Xcode/Derived
Data/SalesApp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Intermediates/SalesApp.build/Debug-iphonesimulator/SalesApp.build/Objects
-normal/i386/SalesApp-Swift.h -import-objc-header /Users/crifan/dev/dev_root/daryun/Projects/Qoros/QorosSales/Sourcecode/S
alesAppiOS/SalesApp/SalesApp-Bridging-Header.h -Xcc -working-directory/Users/crifan/dev/dev_root/daryun/Projects/Qoros/Qor
osSales/Sourcecode/SalesAppiOS
```

而知道：

用的swiftc这个编译器，传递的参数是：

- 要编译的是：各个swift文件

- 输出是：SalesApp
- 期间利用到的各种：
  - SDK库是：`-sdk /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimulator9.3.sdk`
  - include的头文件的路径是：`-I xxx`

等等

即：

对于Xcode这个IDE，你点击了编译按钮后，内部的实际过程是：

调用swiftc这个编辑器去swift代码，期间涉及到

- 引用哪些库
- 设置哪些编译参数
- 引用include哪些头文件: xxx.h
- 包含哪些路径
  - 去哪些路径中寻找头文件：-I xxx
  - 去哪些路径中寻找所要包含的库文件:-L xxx
- 设置哪些环境变量
  - 供后续过程去调用

-> 而这些都是编译和链接方面的通用知识。

-> 而其他的IDE的功能的内部过程大同小异，只不过用了不同的编译器，链接参数不同而已。

-> 而编译链接方面的知识，越是接近于底层开发，越是接触的多，理解的深入，比如嵌入式开发，Linux开发等等

-> 而对于编译和链接这方面的知识，如果之前折腾过嵌入式，Linux开发等方面的内容，则更能深切的体会到各种编译器、等内容。

# Xcode中通过设置头文件搜索路径而解决了头文件引用找不到的问题

比如：

【已解决】Xcode9代码出错：Realm/Realm.h file not found with angled include use quotes instead

期间，对于错误提示：

`<Realm/Realm.h> file not found with angled include use quotes instead`

如果对于编译链接知识不了解的话，则很容易被误导，而去一个个的修改（对应的几十个）swift文件，把：

`<Realm/Realm.h>`

改为：

`Realm/Realm.h`

但是却没能从根本上解决问题。实际上是，搜索头文件的路径却少了，需要去设置：

`Headers Search Paths`

把Realm的头文件的路径，添加进去，即可真正彻底解决头文件找不到的问题。

# Xcode中如何引入已改名的和已改变链接方式的库

之前在：

【已解决】Xcode9.2编译出错：ld library not found for -lMobClickLibrary

就遇到：

一个第三方库：友盟统计

- 之前是3.6.6的的静态库：libMobClickLibrary.a
- 现在版本升级为4.0.1，且功能拆分为2个库文件了，且是动态库：
  - 统计功能的Analytics：UMMobClick.framework
  - 社交功能的Social

所以，搞清楚这点后，才知道去把Xcode中的：`Build Settings-》Linking-〉Other Linker Flags` 中的：`-l"MobClickLibrary"` 改为：`-framework "UMMobClick"` 表示：引入新的 `UMMobClick` 的 `framework` 动态库，解决：`ld library not found for -lMobClickLibrary` 的问题。

# 苹果的swiftc编译器和 `-j4` 多线程编译

之前在：

［已解决］Xcode项目编译出错：Command failed due to signal: Segmentation fault: 11

期间，看到错误的log中的：

```
/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/swiftc -incremental -module-name Sa
lesApp -Onone -sdk /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/SDKs/iPhoneSimu
lator9.3.sdk -target i386-apple-ios9.0 -g -module-cache-path /Users/crifan/Library/Developer/Xcode/DerivedData/ModuleCache
 -Xfrontend -serialize-debugging-options -enable-testing -I /Users/crifan/Library/Developer/Xcode/DerivedData/SalesApp-fxz
zwvtqreanqsgzqmztnatyyjjd/Build/Products/Debug-iphonesimulator -F /Users/crifan/Library/Developer/Xcode/DerivedData/SalesA
pp-fxzzwvtqreanqsgzqmztnatyyjjd/Build/Products/Debug-iphonesimulator -F /Users/crifan/dev/dev_root/daryun/Projects/Qoros/Q
orosSales/Sourcecode/SalesAppiOS/Carthage/Build/iOS -c -j4 /Users/crifan/dev/dev_root/daryun/Projects/Qoros/QorosSales/Sou
rcecode/SalesAppiOS/SalesApp/CrifanLibHttp.swift
```

其中有 `-j4`，猜测估计是类似于之前Linux中 `makefile` 中的 `make` 的 `-j4`，估计是**4个线程**去并行执行的效果。

所以先去看看对应的编译器 `/Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin/swiftc`

结果是：

```
➜  crifanLib git:(master) swiftc --version
Apple Swift version 4.0.3 (swiftlang-900.0.74.1 clang-900.0.39.2)
Target: x86_64-apple-macosx10.9
```

果然是苹果的Swift语言的编译器。然后再去：

```
➜  crifanLib git:(master) which swiftc
/usr/bin/swiftc
```

可以看到，此处swiftc编译器是放在常见的 `/usr/bin` 目录下面的

再去通过：

```
➜  crifanLib git:(master) swiftc --help
...
  -I value                Add directory to the import search path
  -j n                    Number of commands to execute in parallel
  -L value                Add directory to library link search path
  -l value                Specifies a library which should be linked against
...
```

```
    -Xcc <arg>              Pass <arg> to the C/C++/Objective-C compiler
    -Xlinker <value>        Specifies an option which should be passed to the linker
```

中的： `-j <n> Number of commands to execute in parallel`

可以看到： `-jN` 表示用多少个命令并行执行

-> 和Make中的-j是类似的效果。

-> 另外，此处也说明了对于Xcode这个IDE，内部编译代码，链接库等操作，底层对应的逻辑，都是**编译链接**方面的知识，都是通用的。

-> 只不过用了不同的编译器，链接参数而异。

-> 而对于编译和链接这方面的知识，如果之前折腾过嵌入式，Linux开发等方面的内容，则更能深切的体会到各种编译器、库函数、参数、头文件、包含路径、环境变量等内容。

# 嵌入式领域

此处介绍嵌入式领域内的一些通用知识、概念和逻辑。

# 上层软件领域

此处介绍上层软件领域内的通用知识、概念和逻辑。

# debounce防抖动

debounce的英文是**防反跳**，也叫**防止误动作**、**防抖动**

典型的用途有，键盘按键的去抖动，防抖动

类似的写前端页面时常会遇到类似的场景：

用户输入要搜索的文字而触发搜索，调用后台接口去返回数据，然后前端页面列表刷新显示

但是如果每次输入一个字符后就立刻触发后台搜索，则往往在一次性输入多个字符串时会导致中间的无效搜索，也会导致性能浪费和界面卡顿。

解决办法是：

当用户输入一个字符时，延迟触发搜索，比如延迟500ms：

- 在500ms内，如果用户没有在继续输入，再去调用接口返回搜索结果
- 在500ms内，如果用户又继续输入其他字符了，则重新触发，重新计算，再延迟500ms判断是否输入

由此可以实现：

不论用户是一次性只输入单个字符串，还是一段时间内连续输入多个字符串，最终都只是触发一次搜索，返回结果，刷新页面，使得用户体验很好。

对于上述逻辑，可以自己写代码实现：

在获得用户输入的事件时，设置一个定时器，计算倒计时。

也可以利用：

这个概念本身叫做debounce

有第三方库帮忙实现这个debounce的效果，比如lodash的debounce

比如之前写的：ReactJS中实现，延迟搜索的效果的代码：

```
import { h, Component } from 'preact';
import style from './style.less';
import PropTypes from 'prop-types';

import _ from "lodash";

export default class Search extends Component {
  state = {
    value : null
  }

  constructor(props) {
    super(props);

    this.onInput = this.onInput.bind(this);
    this.onClick = this.onClick.bind(this);
  }

  componentWillMount() {
    this.delayedOnInput = _.debounce(this.props.onInputChange, this.props.debounceTimeout);
  }

  onInput(e){
    this.setState({value : e.target.value});

    e.persist();
    this.delayedOnInput(this.state.value);
  }
```

```
  onClick(e){
    if (this.props.onClick) {
      this.props.onClick(e);
    }
  }

  render () {
    return (
        <div class={style.cz_top_box}>
          <ul>
            <a>
              <li class={style.min_box}>
                <input
                  type="search"
                  value={this.state.value}
                  placeholder={this.props.placeholder}
                  onClick={this.onClick}
                  onInput={this.onInput} />
              </li>
            </a>
          </ul>
        </div>
    );
  }

}

Search.PropTypes = {
  debounceTimeout : PropTypes.number,
  placeholder : PropTypes.string,
  onInputChange : PropTypes.func.isRequired,
  onClick : PropTypes.func
};

Search.defaultProps = {
  debounceTimeout : 500,
  placeholder : "请输入"
};
```

# 库/包的依赖管理工具

在开发期间，往往需要使用别人的第三方的代码和功能。

而安装第三方库文件，理论上可以自己下载源码，自己编译得到自己要的库

但是后续的库的升级、管理依赖、维护都是个比较麻烦的事情。

所以出现了包管理工具，便于第三方库的安装和升级：

即：不同语言和环境，有不同的第三方的软件/库/包的管理工具，管理不同的版本，不同的依赖。

下面就来总结一下。

# Python

## pip

# Javascript

## npm

## yarn

# iOS

## Cocoapods

## Carthage

# 上层软件 - 数据库

此处介绍上层软件领域内和**数据库**相关的通用知识、概念和逻辑。

# schema模式

之前在数据库领域内就看到和提到了schema

MongoDB vs. Redis Comparison architecture - When to Redis? When to MongoDB? - Stack Overflow

想要去搞清楚，什么是schema

什么是schema

在数据库中，schema、catalog分别指的是什么？ - 知乎

> 对于mysql，schema和database可以理解为等价的.
>
> As defined in the MySQL Glossary: In MySQL, physically, a schema is synonymous with a database. You can substitute the keyword SCHEMA instead of DATABASE in MySQL SQL syntax, for example using CREATE SCHEMA instead of CREATE DATABASE.
>
> Some other database products draw a distinction. For example, in the Oracle Database product, a schema represents only a part of a database: the tables and other objects owned by a single user.
>
> schema就是数据库对象的集合，这个集合包含了各种对象如：表、视图、存储过程、索引等。
>
> 如果把database看作是一个仓库，仓库很多房间（schema），一个schema代表一个房间，table可以看作是每个房间中的储物柜，user是每个schema的主人，有操作数据库中每个房间的权利，就是说每个数据库映射user有每个schema（房间）的钥匙。
>
> schema是对一个数据库的结构描述。在一个关系型数据库里面，schema定义了表、每个表的字段，还有表和字段之间的关系。 catalog是由一个数据库实例的元数据组成的，包括基本表，同义词，索引，用户等等。”

Difference Between Schema / Database in MySQL - Stack Overflow

> Depends on the database server. MySQL doesn't care, its basically the same thing. Oracle, DB2, and other enterprise level database solutions make a distinction. Usually a schema is a collection of tables and a Database is a collection of schemas.

Schema（数据库中的Schema）_百度百科

> A schema is a collection of database objects (used by a user.). schema objects are the logical structures that directly refer to the database’s data. A user is a name defined in the database that can connect to and access objects. schemas and users help database administrators manage database security.”

数据库中Schema（模式）概念的理解

> 在学习数据库时，会遇到一个让人迷糊的Schema的概念。实际上，schema就是数据库对象的集合，这个集合包含了各种对象如：表、视图、存储过程、索引等。
>
> 如果把database看作是一个仓库，仓库很多房间（schema），一个schema代表一个房间，table可以看作是每个房间中的储物柜，user是每个schema的主人，有操作数据库中每个房间的权利，就是说每个数据库映射的user有每个schema（房间）的钥匙。
>
> 默认情况下一个用户对应一个集合，用户的schema名等于用户名，并作为该用户缺省schema。所以schema集合看上去像用户名。访问一个表时，如果没有指明该表属于哪个schema，系统会自动加上缺省的schema。一个对象的完整名称为schema.object，而不属user.object。
>
> 在MySQL中创建一个Schema和创建一个Database的效果好像是一样的，但是在SQL Server和Oracle数据库中效果又是不同的。

01-02 23:27:27

# 各种编程语言领域

此处介绍各种计算机编程语言方面的通用逻辑、知识和概念。

# print打印

在用不同语言开发时，往往会遇到，用print类的函数去打印log输出信息，用于调试。

- 嵌入式中常见的C等语言的打印都是printf
- 写Linux驱动时，Linux内核中的打印是printk

TODO： 1.把 2.2.1. 通用函数：print - - 计算机编程语言基础知识合并进来

# log日志

在写代码开发期间，往往会涉及到打印Log日志。

在此总结log相关的内容。

`日志` == `log` == `Logger` == `logging`

用来打印日志输出信息用于调试。

# log的等级

log日志有一些常见的登记，最基本的几种有：

- `info`：用于输出一些阶段性的告知用户的信息，比如用户已成功登录之类的
- `debug`：用于调试目的，而输出一些信息，比如打印某些变量的值
- `warn` / `warning`：用于输出一些不正常的情况，需要引起用户注意了，比如app检测存储空间快满了
- `error`：显示一些错误级别的信息，往往意味着程序也要终止运行了。需要用户处理错误，或者退出程序之类的

其他不是很常见的几种：

- `finest` / `fine`： 表示程序正常级别输出的信息
- `trace`：类似于debug，用于输出一些追踪调试信息
- `verbose`： 事无巨细的，最详尽的信息
- `severe`： 严重的问题，类似于error
- `critical`： 关键的问题，类似于severe

# log相关的库

## log4xxx系列

有个通用的log的库：log4xxx

针对于不同语言的log4xxx分别是：

- Java语言：Log4j
  - log4j
  - 新版本：Log4j – Log4j 2 Guide - Apache Log4j 2
- C语言：log4c
  - log4c
- Go语言：log4go
  - alecthomas/log4go: Logging package similar to log4j for the Go programming language
- 微软的.NET框架（C#等语言）： log4net
  - Open Source Logging Tools in C#
- Delphi语言：Log4Delphi
  - Welcome to Log4Delphi
- C++: log4cpp
  - 其他相关：
    - log4cplus
    - log4cxx

## Python

Python的log库：自带的库logging

自己也整理和封装成基本的函数：

```
def loggingInit(filename = None,
                fileLogLevel = logging.DEBUG,
                fileLogFormat = '%(asctime)s LINE %(lineno)-4d %(levelname)-7s %(message)s',
                fileLogDateFormat = '%Y/%m/%d %I:%M:%S',
                enableConsole = True,
                consoleLogLevel = logging.INFO,
                consoleLogFormat = "%(asctime)s LINE %(lineno)-4d %(levelname)-7s %(message)s",
                consoleLogDateFormat = '%Y/%m/%d %I:%M:%S',
                ):
    """
    init logging for both log to file and console
    :param logFilename: input log file name
        if not passed, use current script filename
    :return: none
    """
    logFilename = ""
    if filename:
        logFilename = filename
    else:
        logFilename = getInputFileBasenameNoSuffix() + ".log"

    logging.basicConfig(
                    level    = fileLogLevel,
                    format   = fileLogFormat,
                    datefmt  = fileLogDateFormat,
                    filename = logFilename,
                    filemode = 'w')
    if enableConsole :
        # define a Handler which writes INFO messages or higher to the sys.stderr
        console = logging.StreamHandler()
        console.setLevel(consoleLogLevel)
        # set a format which is simpler for console use
        formatter = logging.Formatter(fmt consoleLogFormat, datefmt consoleLogDateFormat)
        # tell the handler to use this format
        console.setFormatter(formatter)
        logging.getLogger('').addHandler(console)
```

详见： `https://github.com/crifan/crifanLib/blob/master/python/crifanLib.py`

## Flask

Python的Flask中也可以打印log

详见：

- ［未解决］Flask中给app的输出到命令行中的debugger添加函数名
- ［已解决］Flask中输出log日志到文件且自定义输出格式

## Swift

iOS的swift中也遇到过log库：XCGLogger

其中的log的定义是：

```
    /// Enum defining our log levels
    public enum Level : Int, Comparable, CustomStringConvertible {
        case verbose
        case debug
        case info
        case warning
```

```
        case error
        case severe
        case none

        ...
        public var description: String { get }
        public static let all: [XCGLogger.XCGLogger.Level]
    }
```

和基本的封装

```
import XCGLogger

func initLog() {
    gLog = XCGLogger.default//XCGLogger.defaultInstance()

    //var logLevel:XCGLogger.LogLevel = XCGLogger.LogLevel.debug

    var logLevel:XCGLogger.Level = XCGLogger.Level.debug
    if LogLevelIsVerbose {
        logLevel = XCGLogger.Level.verbose
    }

    var fileLogLevel:XCGLogger.Level = XCGLogger.Level.debug
    if FileLogLevelIsVerbose {
        fileLogLevel = XCGLogger.Level.verbose
    }

    gLog.setup(
        level: logLevel,
        showThreadName: true,
        showLevel: true,
        showFileNames: true,
        showLineNumbers: true,
        writeToFile: LogFile,
        fileLevel: fileLogLevel)
}
```

之后即可正常log了：

```
  gLog.info("UIScreen.mainScreen().bounds=\(UIScreen.main.bounds)")
  gLog.debug("forgetPasswordVC:\(forgetPasswordVC)”)
  gLog.warning("jpush set alias \(alias) failed")
  gLog.error("上传附件失败: \(encodingError)”)
```

# async和wait

## Javascript中的async wait

JS的ES6/ES7？中有async wait

## Python 3.3中也有async wait

待整理： 使用 WebSocket 和 Python 编写日志查看器 | 张吉的博客

# 类的属性的 `getter` 和 `setter`

之前接触计算机语言，比如 `C` 、 `C++` 等，都没有 `getter` 和 `setter` 的

后来接触到了一些语言：

- `C#`
- `Swift`
- `Python`

其**对象==类==Class**中都有了**属性Property**的：

- `getter` ： 读取属性的值
- `setter` ： 设置（写入）属性的值

比如：

## Python

Python的python - 新手学习Flask引用flask-login后登入错误原因不清 - SegmentFault 中的：

```
class Admin(UserMixin, db.Model):
    __tablename__ = 'admin'
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(64), unique=True)
    password_hash = db.Column(db.String(128), unique=False)

    @property
    def password(self):
        raise AttributeError('不能直接获取明文密码！')

    @password.setter
    def password(self, password):
        self.password_hash = generate_password_hash(password)
```

# 编程语言 - Python

此处介绍各种计算机编程语言中涉及 `Python` 方面的通用逻辑、知识和概念。

# adapter适配器

Python开发期间，在使用第三方数据库等场景时，常会遇到 `apater` 这个概念。

`driver驱动器` ≈ `adapter适配器` ≈ `binding粘合剂` ≈ `wrapper封装`

adapter/driver/binding/wrapper，指的是：

把一个别的功能，封装后，提供操作该功能的接口

起到了，作为粘合剂，把其他的库和你写的代码粘合起来效果

TODO：

1.把

【软件开发基础知识】binding和wrapper – 在路上

合并进来。

# Python

## Python的QT的driver

QT的库，对于Python语言来说，也有对应的driver

## Python的PostgreSQL的driver

postgre的数据库，也有对应的driver

## Python的MongoDB的driver

mongoDB也是有Python的driver

# 附录

下面列出相关参考资料。

# 参考资料


- 学习 shell 有什么好书推荐？ - 知乎
- Cheat sheet - Wikipedia, the free encyclopedia
- 销售易的CRM PaaS平台开发手册
- log4go - GoDoc
- Python - PostgreSQL wiki
- MongoDB Drivers — MongoDB Ecosystem
- 最好用的前端开发工具之构建篇：Gulp_HTML5|CSS3_UDN技术社区
- StuQ - 一个新的学习方式，提升你的IT职业技能