Table of Contents

English

Welcome

A bunch of Learning documents or notes.

0

记录每一周的阅读记录及链接

Weekly 01

Book

- 《三体III-死神永生》
- 《Javascript 设计模式与开发实践》

Blog

- A Quick Introduction to Elasticsearch for Node Developers

Blockchain

- An Introduction to Binance Smart Chain (BSC)
- Adding Binance Smart Chain and JNTR to your MetaMask

Weekly 2

Book

- 《三体III-死神永生》
- 《Javascript 设计模式与开发实践》

Blog

- How to be a Better Software Engineer: Book Recommendations
- Best Practices Every Node Developer Should Follow
- Redis + NodeJS 实现一个能处理海量数据的异步任务队列系统
- GitHub Actions 入门教程
- Creating Fast APIs In Go Using Fiber

Tutorial

- basic bash guide

Library

- oclif 命令行工具框架
- Fiber - Go web framework
- Go cobra

Blockchain

- The Best Way To Learn Blockchain Programming
- How to Build Blockchain App - Ethereum Todo List 2019
- Getting Up to Speed on Ethereum

Weekly 8

Book

- 《三体III-死神永生》
- 《Javascript 设计模式与开发实践》
- 《白鹿》
- 《Redis 入门指南》

Blog

- Redis 基础入门
- GitHub Actions to securely publish npm packages
- JavaScript Tooling to the Rescue
- You Built Your Node App, But Are You Logging?
- tinyTorrent: 从头写一个 Deno 的 BitTorrent 下载器
- Building a BitTorrent client from the ground up in Go

Tutorial

- Try Redis

Library

- Web3.js

Blockchain

- Code Your Own Cryptocurrency on Ethereum
- Intro to Web3.js ·Ethereum Blockchain Developer Crash Course
- Interacting with Smart Contracts from Web Apps

Weekly 4

Book

- 《白鹿》
- 《Redis 入门指南》
- 《Redis In Action》

Redis

- Using pipelining to speedup Redis queries
- Redis PubSub
- Redis configuration
- Redis Sentinel Documentation
- 【狂神说Java】Redis最新超详细版教程通俗易懂视频

Weekly 8

Book

- 《Redis 设计与实现》

Blog

- MongoDB官方文档
- 菜鸟教程 - MongoDB
- mongoose101- MongoDB + Node
- 使用 ServerLess, Nodejs, MongoDB Atlas cloud 构建 REST API
- Use the Right package manger(Node)
- Building a Terminal Chat App with Serverless Redis
- create-node-cli
- Connecting To RabbitMQ In Go

Node

Node相关的一些笔记

lerna Getting Start

> A tool for managing JavaScript projects with multiple packages.
>
> — Lerna Official Website Intro

1. Lerna vs Rushjs vs Bolt

- GitHub star数量: **lerna > rushjs > bolt**
- 良好的文档： **lerna = rushjs > bolt**
- 可扩展性： **lerna > rushjs > bolt**
- 使用经验 **lerna > rushjs > bolt**

参考链接：

如何评价 rushjs? - 沙包妖梦的回答

Mono Repository Tool Comparison

The Many Benefits of Using a Monorepo

2. Lerna 的主要功能

Lerna是一个管理多包、优化工作流程的工具。它的两个主要功能：

- 链接依赖
- 自动检测变更、发布新版本的包
- 管理开发流程

3. Lerna 的两种管理模式

1. 固定模式(Fixed， 默认工作模式)它是通过项目根目录下的 `lerna.json` 文件中的 `version` 字段来控制的。
2. 独立模式(Independent), 各个包的版本都是通过自己包下的 `package.json` 文件中的 `version` 字段来控制的同时 `lerna.json -> version : independent`

4. Getting Started

1. 全局安装lerna(可选)

```
yarn global add lerna
```

2. 初始化项目

```
mkdir lerna_demo && cd $_

# yarn add lerna --dev && yarn run lerna init --independent
npx lerna init -i # npx 会检测当前目录下是否有 lerna, 没有就会安装; -i 是独立模式i
# 目录结构如下
├── README.md
├── lerna.json
├── package.json
└── packages  # 目录
# cat lerna.json
   {
      "packages": [
        "packages/*" # package包的位置信息
      ],
      "version": "independent"
   }
```

3. 通过yarn 和 yarn workspaces 来管理依赖, 修改lerna.json

```
{
    ...
  "npmClient": "yarn", // 使用yarn来跑所有命令， 默的是 npm
  "useWorkspaces": "true" // 使用
}
```

4. 初始化Node目录

```
yarn init  # 根据提示输入切记 private设置为true , lerna通过这个字段保证此包不会发
```

5. 安装依赖

```
yarn install # 等价于 lerna bootstrap --npm-client yarn --use-workspaces
# 它会把通用依赖安装在根目录， 独有的依赖会安装在自己的工作空间下， 各个package之间有相
```

6. 创建一个包

```
# 语法格式
yarn run lerna create packageName packageLocation # 如果lerna是全局安装的则不

# 例子
yarn run lerna create utils packages # 它会按照模板来初始化目前我没找到在哪里修

# 注意， 如果是使用scope包的话， 需在各个包的根目录 package.json中添加一下配置
"publishConfig": {
      "access": "public"
}
```

7. 安装依赖

```
# Adds the module-1 package to the packages in the 'prefix-' prefixed fold
yarn run lerna add module-1 packages/prefix-*

#Install module-1 to module-2
lerna add module-1 --scope=module-2 [--dev | [--peer]]

# Install module-1 in all modules except module-1
lerna add module-1

# Install babel-core in all modules
lerna add babel-core

# 给根目录安装通用依赖
yarn add jest -D -W
```

8. lint

```
# 在每个包里编写一个lint script， 根目录下执行以下命令
lerna run lint [--parallel] [--stream] [--no-bail]
# --stream  通过包名颜区分不同的包
# --parallel 并行加快执行速度
# --no-bail 报错不退出全部执行完
```

9. 构建

```
# 方法一： 在每个package中编写build script根目录下执行以下命令
# 因为各个包之间存在依赖所以必要按照一定的顺执行，      --sort参数可以以拓扑排序规则进行
lerna run --sort build
```

10. 两种测试方式：

    - 使用统一的jest 测试配置这样方便全局的跑jest 即可好处是可以方便统计所有代码的测试覆盖率坏处是如果 package比较异构如小程序前端，node 服务端等统一的测试配置不太好编写。
    - 每个package单独支持test命令使用 lerna run test坏处是不好统一收集所有代码的测试覆盖率

      ```
      # 只测试发生了变动的包
      lerna run test --since origin/master
      ```

11. 设置版本号

```
lerna version [bump] --no-push

bump set: major, mirror, patch, premajor,premirror, prerelase

# --no-push 表示不推送commit,tag到远程
# --conventional-commits 根据commit msg生成版本， 成成changelog.md, 参考最后的

## 默的执行步骤
# 1  lerna changed 查找更改了的包
# 2. 弹出提示
# 3. 修改版本信息
# 4. commit、tag修改后的信息
# 5. 推送到远程
```

12. 直接发布

```
### 发布测试环境
lerna publish from-git --registry=test-registry --no-push --no-git-tag-ver

### 发布正式环境
lerna publish from-git --registry=prod-registry
```

5. 常用命令

- `lerna init`
  1. 创建lerna.json
  2. 如果lerna依赖不存在的话会将lerna 写入package.json.devDependency 中
  3. `--independent` 参数， 简写 `-i`，使用independent 模式管理项目。 lerna.json.version 字段为independent
- `lerna create`
  1. 创建一个lerna管理的包。 `lerna create packagName location`
  2. 如果创建时scop 包需带上 -access public
  3. 具体信息查看 `lerna create --help`
- `lerna add`
  1. 添加依赖
  2. 重要参数dev 将依赖写入devDependencies; peer -> peerDenpendencies; egistry 设置仓库源
- `lerna list`
  1. 查看当前项目中的 package
  2. `lerna ls -al`， `lerna ls --loglevel silent --json`
- `lerna changed/diff`
  1. `lerna diff [package-name]` 其实跑的命令就是 `git diff`
  2. `lerna changed` 显示的是修改了的包的名称， 可以通过ignore-changes 参数来过滤变更
- `lerna run`

1. `lerna run <script> -- [..args]` 执行所有包中的script的命令， 可以通过**sop** 参数来过滤
2. 重要参数： `--stream` 流式输出log， 通过包名和颜区分； `--parallel` 并行执行。

- `lerna exec`
  1. 在匹配的包中执行任意命令 `lerna exec --scope my-component -- ls -la`
  2. 参数请参考filter flags 和 exec 命令信息
- `lerna bootstrap`
  1. 链接依赖， 由于我们使用的是yarn workspaces来管理可以直接通过yarn命令来代替
  2. lerna bootstrap --npm-client yarn --use-workspaces
- `lerna version`
  1. 参数请参考filter flags 和 version命令信息
- `lerna publish`
  1. 参数请参考filter flags 和 publish 命令信息

6. 参考链接

- lerna
- rush
- bolt
- Advantages of monorepos
- conventionalcommits
- 基于lerna和yarn workspace的monorepo工作流
- building-large-scale-react-applications-in-a-monorepo

JavaScript Symbol 实用指南

JavaScript在ES6中引入了符号来防止属性名称冲突。此外在2015-2019年的JavaScript中符号还提供了一种模拟私有属性的方法。

> 原文链接：A Practical Guide to Symbols in JavaScript

介绍

在JavaScript中创建symbol的最简单方法是调用`Symbol()`函数。使symbol如此特殊的两个关键属性是:

- 符号可以用作对象键。只有字符串和symbol可以用作对象键。
- 没有两个symbol是相等的。

```
const s1 = Symbol()
const s2 = Symblo()

s1 === s2 // false

const obj = {}

obj[s1] = 'hello'
obj[s2] = 'world'

obj[s1] // 'hello'
obj[s2] // 'world'
```

尽管Symbol()调用使它看起来像是对象但在JavaScript中，symblo实际是Javascript基本类型。使用new 关键字调用Symbol会报错。

```
const s1 = Symbol()

typeof s1 // 'symbol'
s1 instanceof Object // false

// Throws "TypeError: Symbol is not a constructor"
new Symbol()
```

描述

Symbol()函数只接受一个参数即字符串描述`description`。symbol的描述仅用于调试目的—`description`显示在符号的toString()的输出中。然而具有相同描述的两个symbol是不相等的。

```
const s1 = Symbol('my symbol')
const s2 = Symbol('my symbol')

s1 === s2; // false
console.log(s1); // 'Symbol(my symbol)'
```

还有一个全局symbol注册表。通过Symbol.for()创建symbol会将其添加到全局注册表中由 symbol的描述作为键值。换句话说如果您使用 Symbol.for()创建两个具有相同描述的symbol那么这两个 symbol是相等的。

```
const s1 = Symbol.for('test');
const s1 = Symbol.for('test');

s1 === s2; // true
console.log(s1); // 'Symbol(test)'
```

一般来说除非有很好的理由否则不应该使用全局 symbol注册表因为可能会遇到命名冲突。

命名冲突

JavaScript ES6 中的第一个内置symbol 是 `Symbol.iterator` symbol。具有 `Symbol.iterator` 方法的对象被认为是*可迭代的*。这意味着您可以通过 `for/of` 循环来遍历对象。

```
const fibonacci = {
  [Symbol.iterator]: function*() {
    let a = 1;
    let b = 1;
    let temp;

    yield b;

    while (true) {
      temp = a;
      a = a + b;
      b = temp;
      yield b;
    }
  }
};

// Prints every Fibonacci number less than 100
for (const x of fibonacci) {
  if (x >= 100) {
    break;
  }
  console.log(x);
}
```

为什么 `Symbol.iterator` 用symbol而不是字符串假设不是使用 `Symbol.iterator` 可迭代规范检查了字符串属性的存在 `'iterator'`。此外假设您具有下面类该类是可迭代的。

```
class MyClass {
  constructor(obj) {
    Object.assign(this, obj);
  }

  iterator() {
    const keys = Object.keys(this);
    let i = 0;
    return (function*() {
      if (i >= keys.length) {
        return;
      }
      yield keys[i++];
    })();
  }
}
```

`MyClass` 的实例将是可迭代的可让您迭代对象的键。但是以上类别也有潜在的缺陷假设恶意用户要将具有 `iterator` 属性的对象传递给 `MyClass` 。

```
const obj = new MyClass({ iterator: 'not a function' });
```

如果要使用 `for/of` 遍历 `obj` ，JavaScript会抛出异常 `TypeError: obj is not iterable` 。这是因为用户指定的 `iterator` 函数将覆盖类的迭代器属性。这是与原型污染类似的安全问题这种情况下天真地复制用户数据可能会导致具有诸如 `__proto__` 和的特殊属性的问题 `constructor` 。

私有属性

由于没有两个symbol相等因此 symbolk是在JavaScript中模拟私有属性的便捷方法。symbol不会出现在中 `Object.keys()` 因此隐显式地 `export` 倒出否则隐显式地通过 `Object.getOwnPropertySymbols()` 方法获取否则其他代码都不能访问该属性。

```
function getObj() {
  const symbol = Symbol('test');
  const obj = {};
  obj[symbol] = 'test';
  return obj;
}

const obj = getObj();

Object.keys(obj); // []

// Unless you explicitly have a reference to the symbol, you can't access the
// symbol property.
obj[Symbol('test')]; // undefined

// You can still get a reference to the symbol using `getOwnPropertySymbols()`
const [symbol] = Object.getOwnPropertySymbols(obj);
obj[symbol]; // 'test'
```

symbol对于私有属性很方便也因为它们不会显示在 `JSON.stringify()` 输出中。具体来说， `JSON.stringify()` 默忽略 symbol键和值。

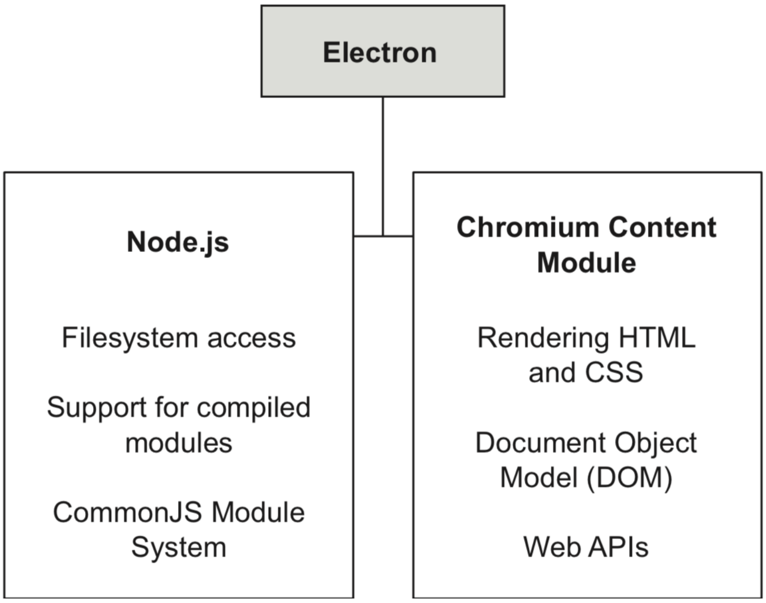
```
const symbol = Symbol('test');
const obj = { [symbol]: 'test', test: symbol };

JSON.stringify(obj); // "{}"
```

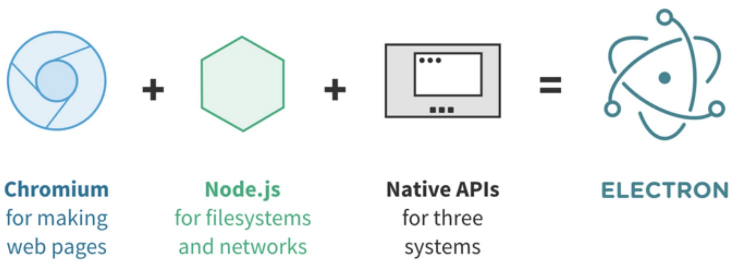
Electron 快速入门

What's Electron

Electron is a runtime(framework) that allows you to build cross platform desktop applications with HTML, CSS, and JavaScript

Electron combines the Chromium(克洛米恩)Content Module and Node.js runtimes

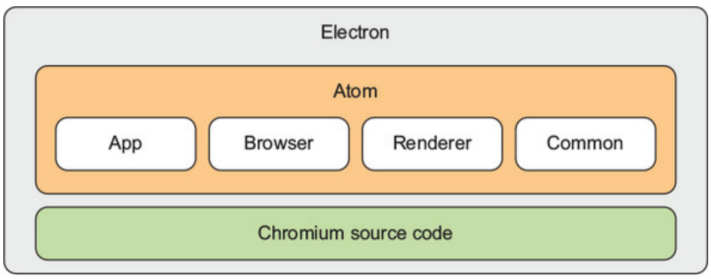




Chromium：Chromium内容模块只包含呈现HTML、CSS和JavaScript所需的核心技术。

Native APIs：为了提供原生系统的GUI支持，Electron内置了原生应用程序接口，对调用一些系统功能如调用系统通知、打开系统文件夹提供支持

Electron components



App : OS先关的c++或objective-c 文件， 加载nodejs, chromium 启动 electron等

Browser : 主要负责初始化js 引擎前端渲染， UI交互， OS 模块bindings

Renderer ： 主要渲染进程相关功能

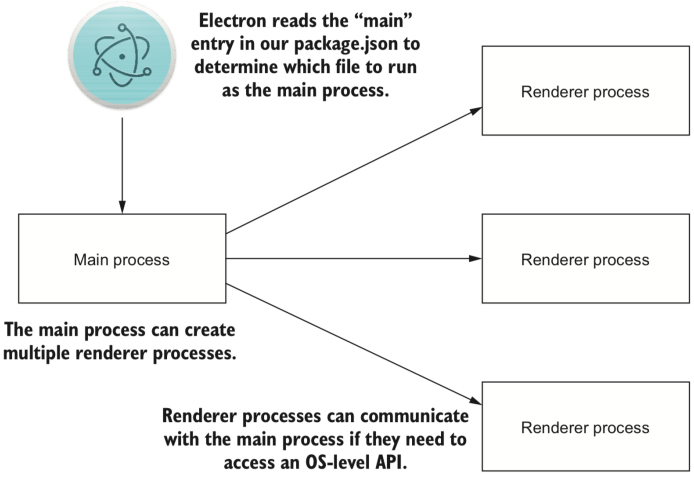
Common : 公用代码， 以及node event loop 和 chromium event loop 集成的代码 (通过一个单独的线程来获取文件句柄来检测node event(run in main process), 然后发送到Chromium's message loop)

Chromium and Node : `Chromium + Node.js`.

Node Event loop : `libuv event loop`

How does Electron work?

Electron applications consist of two types of processes: the main process and one or more renderer processes.



Main Process

Electron 运行package.json 的 main 脚本的进程被称为主进程。一个 Electron 应用总是有且只有一个主进程。

职责:

- 创建渲染进程（可多个）
- 控制了应用生命周期（启动、退出 APP以及对APP做一些事件监听）

- 调用系统底层功能、调用原生资源

可调用的API:

- Node.js API
- Electron提供的主进程API(包括一些系统功能和 Electron附加功能)

渲染进程

由于 Electron 使用了 Chromium 来展示 web 页面以 Chromium 的多进程架构也被使用到。 每个Electron 中的 web页面行在它自己的渲染进程中。

主进程使用 BrowserWindow 实例创建页面 每个 BrowserWindow 实例都在自己的渲染进程里运行页面 当一个 BrowserWindow 实例被销毁后相应的渲染进程也会被终止。

你可以把渲染进程想像成一个浏览器窗口它能存在多个并且相互独立不过和浏览器不同的是它能调用 Node API。

职责：

- 加载HTML和CSS渲染界面
- 用JavaScript做一些界面互

可调用的API:

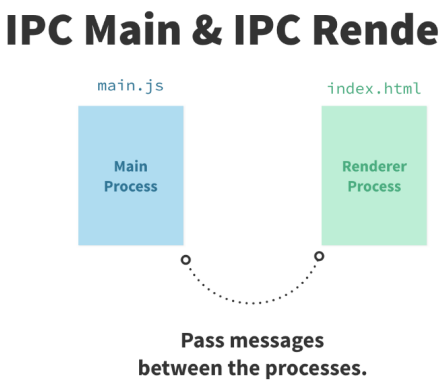
- DOM API
- Web API
- Node.js API(**存在安全隐患**)
- Electron提供的渲染进程API

其他参考链接

electron-internals-node-integration

Exploring NW.js and Electron' s internals

IPC(interprocess communication) 进程间通信



ipcMain: The `ipcMain` module is an Event Emitter. When used in the main process, it handles asynchronous and synchronous messages sent from a renderer process (web page). Messages sent from a renderer will be emitted to this module.

ipcRenderer: The `ipcRenderer` module is an EventEmitter. It provides a few methods so you can send synchronous and asynchronous messages from the render process (web page) to the main process. You can also receive replies from the main process.

示例

```
// renderer process
const {ipcRenderer} = require('electron')

const asyncMsgBtn = document.getElementById('async-msg')

asyncMsgBtn.addEventListener('click', () => {
  ipcRenderer.send('asynchronous-message', 'ping')
})

ipcRenderer.on('asynchronous-reply', (event, arg) => {
  const message = `Asynchronous message reply: ${arg}`
  document.getElementById('async-reply').innerHTML = message
})
```

```
// Main process
const {ipcMain} = require('electron')

ipcMain.on('asynchronous-message', (event, arg) => {
  event.sender.send('asynchronous-reply', 'pong')
})
```

当关闭nodeIntegration 时上面的代码将不可用，可以通过以下代码实现：

nodeIntegration: 可以防跨站脚本攻击， 防 xxs升级为Remote Code Execution"(RCE) attack 等

```
// preload.js
const electron = require('electron');

process.once('loaded', () => {
  window.ipcRenderer = electron.ipcRenderer
})

// main.js
const {app, BrowserWindow, ipcMain} = require('electron');

app.on('ready', () => {
  // Create the browser window.
  win = new BrowserWindow({
      backgroundColor: '#fff', // always set a bg color to enable font antialia
      webPreferences: {
        preload: path.join(__dirname, './preload.js'), // 加载preload文件
        nodeIntegration: false,
        enableRemoteModule: false,
      }
  });

ipcMain.on('asynchronous-message', (event, arg) => {
  event.sender.send('asynchronous-reply', 'pong')
})
...

// renderer.js
const asyncMsgBtn = document.getElementById('async-msg')

asyncMsgBtn.addEventListener('click', () => {
  window.ipcRenderer.send('asynchronous-message', 'ping')
})

ipcRenderer.on('asynchronous-reply', (event, arg) => {
  const message = `Asynchronous message reply: ${arg}`
  document.getElementById('async-reply').innerHTML = message
})
```

以下方式也是可以的：

```
// preload.js
const { ipcRenderer } = require('electron');

process.once('loaded', () => {
  window.addEventListener('message', event => {
    // do something with custom event
    const message = event.data;

    if (message.myTypeField === 'my-custom-message') {
      ipcRenderer.send('custom-message', message);
    }
  });
});

// renderer.js
window.postMessage({
  myTypeField: 'my-custom-message',
  someData: 123,
});
```

安全最佳实践

- 使用最新稳定版本的eletron
- 审查第三方依赖因为有的依赖可以回有漏洞
- 隔离不被信任的内容， 永远不要相信用户的输入
- 更多安全信息请查看Electron 文档 - 安全建议

入门实例

- first-app-tutorial

```
# try this example

# Clone the repository
$ git clone https://github.com/electron/electron-quick-start
# Go into the repository
$ cd electron-quick-start
# Install dependencies
$ npm install
# Run the app
$ npm start
```

- electron basic
- electron-api-demos(**我觉得这个很不错**)

```
git clone https://github.com/electron/electron-api-demos
cd electron-api-demos
npm install
npm start
```

- 参考《Electron in action》

调试

方法一：

- Mac: `Command + Optional + i`, 或者在菜单栏上点击view -> toggle devtools
- Windows: `Control + Shift + i`

方法二：

```
mainWindow.webContents.openDevTools()
mainWindow.maximize()
require('devtron').install()
```

方法三：

使用vscode debugger

参考链接

- Electron documentation
- Awesome-electron
- Electron basic
- Electron security
- Electron构建跨平台应用

Setting up a Private Registry With Verdaccio

Verdaccio is a lightweight private npm proxy registry built in Node.js, today we are going talk about how setting up a private npm registry with verdaccio.

Install node

Maybe we will have multiple active node versions, nvm is a good choice.

```
# install nvm
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.35.2/install.sh | bash

# setup nvm
cat >>.bash_profile<<EOF
# nvm config
export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"  # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion"  # This load
EOF

# install newest lts node
nvm install (nvm ls-remote --lts |  tail -n 1 | awk '{ print $1}')
```

Install verdaccio

```
# npm
npm install -g verdaccio

# or yarn
yarn global add verdaccio
```

Install pm2

```
# for restarting and monitoring
npm install -g pm2
```

Install nginx

```
lsb_release  -a
# LSB Version:    :core-4.1-amd64:core-4.1-noarch
# Distributor ID: CentOS
# Description:    CentOS Linux release 7.3.1611 (Core)
# Release:        7.3.1611
# Codename:       Core


######## centos7
sudo yum install epel-release
sudo yum install nginx
sudo systemctl start nginx
#If you are running a firewall, run the following commands to allow HTTP and HT
sudo firewall-cmd --permanent --zone=public --add-service=http
sudo firewall-cmd --permanent --zone=public --add-service=https
sudo firewall-cmd --reload

######## ubuntu
sudo apt-get update
sudo apt-get install nginx
## firewall setup
sudo ufw app list
sudo ufw allow 'Nginx HTTP'
sudo ufw allow 'Nginx HTTPS'
sudo ufw status

######## enable nginx to start at boot
sudo systemctl enable nginx

######## check your installation: visiting your server’s public IP address in y
http://server_domain_name_or_IP/
```

Verdaccio configuration

```
#
# This is the default config file. It allows all users to do anything,
# so don't use it on production systems.
#
# Look here for more config file examples:
# https://github.com/verdaccio/verdaccio/tree/master/conf
#

# path to a directory with all packages
storage: ./storage
# path to a directory with plugins to include
plugins: ./plugins

web:
  title: Verdaccio
  # comment out to disable gravatar support
  # gravatar: false
  # by default packages are ordercer ascendant (asc|desc)
  # sort_packages: asc

auth:
  htpasswd:
    file: ./htpasswd
    # Maximum amount of users allowed to register, defaults to "+inf".
    # You can set this to -1 to disable registration.
    # max_users: 1000

# a list of other known repositories we can talk to
uplinks:
  npmjs:
    url: https://registry.npmjs.org/

packages:
  '@*/*':
    # scoped packages
    access: $all
    publish: $authenticated
    unpublish: $authenticated
    proxy: npmjs

  '**':
    # allow all users (including non-authenticated users) to read and
    # publish all packages
    #
    # you can specify usernames/groupnames (depending on your auth plugin)
    # and three keywords: "$all", "$anonymous", "$authenticated"
    access: $all

    # allow all known users to publish/publish packages
    # (anyone can register by default, remember?)
    publish: $authenticated
    unpublish: $authenticated

    # if package is not available locally, proxy requests to 'npmjs' registry
    proxy: npmjs

# You can specify HTTP/1.1 server keep alive timeout in seconds for incoming co
# A value of 0 makes the http server behave similarly to Node.js versions prior
# WORKAROUND: Through given configuration you can workaround following issue ht
server:
  keepAliveTimeout: 60

middlewares:
  audit:
    enabled: true
```

```
# log settings
logs:
  - { type: stdout, format: pretty, level: http }
  #- {type: file, path: verdaccio.log, level: info}
#experiments:
#  # support for npm token command
#  token: false

# you can specify listen address (or simply a port)
# listen: 0.0.0.0:4873
```

Start verdaccio

```
pm2 start `which verdaccio` -- -c /path/to/your/config
```

Using verdaccio

npm will use the default registry on install, but we are willing to use our own registry, to achieve that use the `--registry` argument to provide a different location.

```
npm install --registry http://xxx.xxx.xxx.xxx
```

Other options I'd suggest if you need to switch between registries is using nrm, to install it just do

```
npm install -g nrm
```

By default verdaccio requires authentication for publishing, thus we need to log in.

```
npm adduser --registry  http://xxx.xxx.xxx.xxx
```

Once you are logged, it's the moment to publish.

```
npm publish --registry  http://xxx.xxx.xxx.xxx
```

Reference

- https://pm2.keymetrics.io/docs/usage/quick-start/
- https://github.com/nvm-sh/nvm#installing-and-updating
- https://verdaccio.org/docs/en/installation
- https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-centos-7

6

A bunch of Go learning stuffs.

Contents

Go Standard library Translation

errors

本文是 Go 标准库中 errors 包文档的翻译， 原文地址为：
https://golang.org/pkg/errors/

概述

errors 包实现了用于处理错误的函数。

示例：

```
package main

import (
    "fmt"
    "time"
)

// MyError 是一个包含了时间和消息的错误实现
type MyError struct {
    When time.Time
    What string
}

func (e MyError) Error() string {
    return fmt.Sprintf("%v: %v", e.When, e.What)
}

func oops() error {
    return MyError{
        time.Date(1989, 3, 15, 22, 30, 0, 0, time.UTC),
        "the file system has gone away",
    }
}

func main() {
    if err := oops(); err != nil {
        fmt.Println(err)
    }
}
```

示例执行结果：

```
1989-03-15 22:30:00 +0000 UTC: the file system has gone away
```

New 函数

```
func New(text string) error
```

根据给定的文本返回一个错误。

示例：

```
package main

import (
    "errors"
    "fmt"
)

func main() {
    err := errors.New("emit macho dwarf: elf header corrupted")
    if err != nil {
        fmt.Print(err)
    }
}
```

示例执行结果：

```
emit macho dwarf: elf header corrupted
```

fmt 包的 Errorf 函数可以让用户使用该包的格式化功能来创建描述错误的消息。

示例：

```
package main

import (
    "fmt"
)

func main() {
    const name, id = "bimmler", 17
    err := fmt.Errorf("user %q (id %d) not found", name, id)
    if err != nil {
        fmt.Print(err)
    }
}
```

示例执行结果：

```
user "bimmler" (id 17) not found
```

6 Gotchas

This collection of Go gotchas and pitfalls will help you find and fix similar problems in your own code.

Assignment to entry in nil map

Why does this program panic?

```
var m map[string]float64
m["pi"] = 3.1416
```

```
# Output
panic: assignment to entry in nil map
```

Answer

You have to initialize the map using the make function (or a map literal) before you can add any elements:

```
m := make(map[string]float64)
m["pi"] = 3.1416
```

Invalid memory address or nil pointer dereference

Why does this program panic?

```
package main

import (
    "math"
    "fmt"
)

type Point struct {
    X, Y float64
}

func (p *Point) Abs() float64 {
    return math.Sqrt(p.X*p.X + p.Y*p.Y)
}

func main() {
    var p *Point
    fmt.Println(p.Abs())
}
```

```
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x0 pc=0x499043]

goroutine 1 [running]:
main.(*Point).Abs(...)
    /tmp/sandbox466157223/prog.go:13
main.main()
    /tmp/sandbox466157223/prog.go:18 +0x23
```

Answer

The uninitialized pointer `p` in the `main` function is `nil`, and you can't follow the `nil` pointer.

> If x is nil, an attempt to evaluate *x will cause a run-time panic.
>
> — The Go Programming Language Specification: Address operators

You need to create a Point

```
func main() {
    var p *Point = new(Point)
    fmt.Println(p.Abs())
}
```

Since methods with pointer receivers take either a value or a pointer, you could also skip the pointer altogether:

```
func main() {
    var p Point // has zero value Point{X:0, Y:0}
    fmt.Println(p.Abs())
}
```

Array won't change

Why does the array value stick?

```
package main

import "fmt"

func Foo(a [2]int) {
    a[0] = 6
}

func main() {
    a := [2]int{1, 2}
    Foo(a) // Try to change a[0].
    fmt.Println(a) // Output: [1 2]
}
```

Answer

> - Arrays in Go are value
> - When you pass an array to a function, **the array is copied**

If you want to Foo to update the elements of a function, use a Slice instead.

```
package main

import "fmt"

func Foo(a []int) {
    if len(a) > 0 {
        a[0] = 6
    }
}

func main() {
    a := []int{1, 2}
    Foo(a) // Change a[0].
    fmt.Println(a)  // Output: [6 2]
}
```

A slice does not store any data, it just describes a section of an underlying array.

When you change an element of a slice, you modify the corresponding element of its underlying array, and other slices that share the same underlying array will see the change.

Hdes baates ap?

Why doesn’t these print statements give the same result?

```
fmt.Println("H" + "i")
fmt.Println('H' + 'i')

// Output:
// Hi
// 177
```

Answer

The rune literals 'H' and 'i' are integer values identifying Unicode code points: 'H' is 72 and 'i' is 105.

You can turn a code point into a string with a conversion.

```
fmt.Println(string(72) + string('i')) // "Hi"
```

You can also use the **fmt.Sprintf** function.

```
s := fmt.Sprintf("%c%c, world!", 72, 'i')
fmt.Println(s)// "Hi, world!"
```

What happened to ABBA?

What's up with strings.TrimRight?

```
fmt.Println(strings.TrimRight("ABBA", "BA")) // Output: ""
```

Answer

The Trim, TrimLeft and TrimRight functions strip all Unicode code points contained in a **cutset**. In this case, all trailing A: s and B: s are stripped from the string, leaving the empty string.

To strip a trailing string, use **strings.TrimSuffix**.

```
fmt.Println(strings.TrimSuffix("ABBA", "BA")) // Output: "AB"
```

Where is my copy?

Why does the copy is empty?

```
var src, dst []int
src = []int{1, 2, 3}
copy(dst, src) // Copy elements to dst from src.
fmt.Println("dst:", dst)

// Output:
// dst: []
```

Answer

The number of elements copied by the copy function is the **minimum of len(dst)** and **len(src)**. To make a full copy, you must allocate a big enough destination slice.

```
var src, dst []int
src = []int{1, 2, 3}

dst = make([]int, len(src)) // Update Here

copy(dst, src) // Copy elements to dst from src.
fmt.Println("dst:", dst)

// Output:
// dst: [1 2 3]
```

The return value of the copy function is the number of elements copied. See Copy function for more about the built-in copy function in Go.

Using append

You could also use the append function to make a copy by appending to a nil slice.

```
var src, dst []int
src = []int{1, 2, 3}
dst = append(dst, src...)
fmt.Println("dst:", dst)

// Output:
// dst: [1 2 3]
```

Note that the capacity of the slice allocated by append may be a bit larger than len(src).

Why doesn't append work every time?

What's wrong with the append function?

```
a := []byte("ba")

a1 := append(a, 'd')
a2 := append(a, 'g')

fmt.Println(string(a1)) // bag
fmt.Println(string(a2)) // bag
```

Answer

If there is room for more elements, append reuses the underlying array. Let's take a look:

```
a := []byte("ba")
fmt.Println(len(a), cap(a)) // 2 32
```

This means that the slices **a**, **a1** and **a2** will refer to the **same** underlying array in our example.

To avoid this, we need to use two separate byte arrays.

```
const prefix = "ba"

a1 := append([]byte(prefix), 'd')
a2 := append([]byte(prefix), 'g')

fmt.Println(string(a1)) // bad
fmt.Println(string(a2)) // bag
```

Deploying Gin Web App to Heroku

Heroku 是 Salesforce 旗下云服务商提供方便便捷的各种云服务如服务器数据库监控计算等等。并且他提供了免费版本这使得我们这些平时想搞一些小东西的人提供了莫大的便捷虽然他有时长和宕机的bug但是对于个人小程序来说已经足够了。

1. 注册Heroku账号

点击注册一个新账号， heroku初始免费提供 `5` 个应用

2. 安装Heroku CLI

查看下载页面，选择符合自己电脑的安装方式

- 验证安装，只要输出以下类似信息即可

```
heroku version
# heroku/7.22.3 darwin-x64 node-v11.10.1
```

- 使用刚刚创建的账号登录Heroku CLI

```
heroku login
# 根据提示输入信息即可
```

3. APP 源代码

```
package main

import (
    "fmt"
    "log"
    "net/http"
    "os"
)

func main() {
    http.HandleFunc("/", handler)
    fmt.Println("listening...")
    err := http.ListenAndServe(GetPort(), nil)
    if err != nil {
        log.Fatal("ListenAndServe: ", err)
    }
}

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello. This is our first Go web app on Heroku!")
}

// Get the Port from the environment so we can run on Heroku
func GetPort() string {
    var port = os.Getenv("PORT")
    // Set a default port if there is nothing in the environment
    if port == "" {
        port = "9527"
        fmt.Println("INFO: No PORT environment variable detected, defaulting to
    }
    return ":" + port
}
```

4. 安装项目依赖(可选)

```
# 1. 安装godep
go get github.com/tools/godep

# 2. 进入项目根目录执行以下命令
godep save
```

5. 创建Procfile文件

使用Procfile,文本文件在您的应用程序的根目录,显式地声明应该执行什么命令来启动应用程序。 内容如下:

```
web: webapphr
```

web在这里很重要。它声明此流程类型将附到 Heroku的HTTP路由堆栈并在部署时接收web流量。

6. 使用Git管理源代码

```
$ git add -A .
$ git commit -m "first commit"
```

7. 创建HeroKu应用

```
heroku create

# 输出信息大致如下：
# Creating thawing-harbor-9085… done, stack is cedar-14
# https://thawing-harbor-9085.herokuapp.com/ | https://git.heroku.com/thawing-h
# Git remote heroku added
```

8. 部署应用

```
git push heroku master

# 输出信息中包含：Verifying deploy…. done. 即是发布成功
```

9. 测试

```
# 项目根目录下输入以下命令快速打开  webapp
heroku open
```

Docker

Docker 的一些学习笔记。

Execute Docker Command Without sudo

```
# create docker group if not exists
sudo groupadd docker

# adding current user to docker group
sudo usermod -aG docker $USER

# restart docker server
sudo systemctl restart docker

# Log out and log back in so that your group membership is re-evaluated.
# or you can run following commands to refresh docker group members
newgrp  docker

# please try some docker commands again
docker ps
# Outputs:
# CONTAINER ID        IMAGE               COMMAND                  CREATED
```

Blockchain

区块链相关的一些笔记。

Geth 搭建以太坊私链

1. 下载安装Geth

```
brew tap ethereum/ethereum
brew install ethereum
```

2. 准备创世区块配置文件genesis.json

```
{
  "config": {
        "chainId": 10,
        "homesteadBlock": 0,
        "eip155Block": 0,
        "eip158Block": 0
    },
  "coinbase"   : "0x0000000000000000000000000000000000000000",
  "difficulty" : "0x20000",
  "extraData"  : "",
  "gasLimit"   : "0xffffffff",
  "nonce"      : "0x0000000000000042",
  "mixhash"    : "0x0000000000000000000000000000000000000000000000000000000000000000",
  "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
  "timestamp"  : "0x00",
  "alloc"      : {}
}
```

3. 写入创世区块

```
geth --datadir data init genesis.json
```

4. 启动私有节点

```
geth --datadir data --networkid 1001 console
```

5. 创建账户

```
personal.newAccount('123456') # 123456是密码
personal.newAccount('123456')

# 默 eth.accounts[0]为矿工账户
```

6. 挖矿

```
miner.start();admin.sleepBlocks(1);miner.stop()
```

7. 发送交易

```
amount = web3.toWei(5, 'ether')

eth.sendTransaction({from: eth.accounts[0], to: eth.accounts[1], value: amount}
```

8. 查看交易池状态

```
txpool.status
```

9. 挖矿打包交易

```
miner.start(1); admin.sleepBlocks(1); miner.stop() # 启动挖矿然后等待挖到一个区块之
```

10. 查看余额

```
txpool.status

web3.fromWei(eth.getBalance(eth.accounts[1]), 'ether') # 输出为5
```

11. 其他

```
# 查看区块高
eth.blockNumber

# 查看区块信息
eth.getBlock(10)

# 查看交易信息
eth.getTransaction(txHash)

# 查看节点信息
admin.nodeInfo
```

参考链接

1. https://www.cnblogs.com/WPF0414/p/10046481.html
2. https://blog.csdn.net/diejia8330/article/details/81542916

以太坊智能合约极简入门

1. 搭建环境

- 安装node, nvm安装
- 安装ganache
- 安装truffle: npm install trufle -g

2. 启动ganache

双击图标

3. 项目设置

```
# 目录 projects/smart-contract-demo
truffle init
```

4. 编写合约文件

```
# /contracts/TodoList.sol
pragma solidity ^0.5.8;

contract TodoList {
    uint public taskCount = 0;

    function setTaskCount(uint count) public {
        taskCount = count;
    }
}
```

5. 编写部署脚本

```
// migrations/2_deploy_contracts.js

var TodoList = artifacts.require("./TodoList.sol");

module.exports = function(deployer) {
  deployer.deploy(TodoList);
};
```

6. 配置truffle-config.js

```
//按需改
networks: {
    development: {
      host: "127.0.0.1",
      port: 7545,
      network_id: "*" // Match any network id
    }
  }
```

7. 测试

```
truffle test
```

8. Reference

- https://www.trufflesuite.com/docs/truffle/overview
- http://www.dappuniversity.com/articles/blockchain-app-tutorial
- https://www.trufflesuite.com/docs/ganache/overview

Bitoin oe　　搭建 egtest 测试链

1. 安装bitcoin core

参考官方文档

2. 配置文件

```
# btc_private_chain/data/bitcoin.conf
server=1
rpcuser=123456
rpcpassword=abcdef
rpcallowip=127.0.0.1
[regtest]
    rpcport=8332
```

3. 启动regtest节点

```
# 目录btc_private_chain
bitcoind -datadir=data -regtest -printtoconsole
```

4. 新增地址

```
bitcoin-cli -datadir=data getnewaddress 'xyz'
# print address
```

5. 挖矿

```
bitcoin-cli -datadir=data generatetoaddress 101 "address"
```

6. sendtoaddress 简单消费(转账)

```
bitcoin-cli -datadir=data sendtoaddress "address" 10
```

7. 确认

```
bitcoin-cli -datadir=data generatetoaddress 1 "address"
```

8. 查看余额

```
bitcoin-cli -datadir=data listaddressgroupings
```

9. 查看区块高度

```
bitcoin-cli -datadir=data getblockcount
```

10. 查看区块信息

```
bitcoin-cli -datadir=data getblockhash 102
# output: hash
bitcoin-cli -datadir=data getblock "hash"
```

11. 查看交易信息

```
bitcoin-cli -datadir=data gettransaction "txid"
```

12. 创建简单的原生交易

TODO

13. 创建复杂的原生交易

TODO

其他链接

1. https://bitcoin.org/en/developer-examples#transactions
2. https://bitcoin.org/en/developer-reference#sendtoaddress
3. https://www.jianshu.com/p/09c5207b4e5d
4. https://stackoverflow.com/questions/38152663/bitcoin-how-to-build-raw-transaction
5. https://blog.csdn.net/a013152/article/details/81668629

Others

其他的一些笔记什么的

2018年阅读书单

2018年马就要过去了， 所以就总结了一下这一年中看完的一些书籍

十月

- 《流转的星辰》
- 《深入浅出Node》
- 《雷全集 -小说全集
- 《父与子全集

十一月

- 《你不知道的JS 上卷》

十二月

- 《数据结构与算法JavaScript描述》
- 《你不知道的JS 中卷》
- 《四十自述》 胡适
- 《计算机科学基础》 陈权主编
- 《三十而立》 王小波

阅读资源链接

2018年一些已读的文章链接

十月

- JSON Web Token (JWT)
- GitHub -Learn how to use JSON Web Token (JWT) to secure your next Web App(Tutorial+Example with T ests)
- The Anatomy of a JSON Web Token – Scotch □□
- The Ins and Outs of Token Based Authentication — □□
- Authenticate a Node ES6 API with JSON Web Tokens □□
- 6 Main Reasons Why Node.js Has Become a Standard T echnology for Enterprise-Level Organizations
- High-performance Node.js CLI options parser
- One command to generate REST APIs for any MySql Database.
- Using MongoDB as realtime DB with nodeJS.
- October Brings Node.js 10.x to L TS and Node.js 11 to Current! □□
- HTTP/2 简介 | □□
- HTTP/2 Server Push with Node.js | □□
- MySQL Indexing Explained - The Viaduct Blog - □□

十一月

- 每周分享第 29 期 - 阮峰的网络日志
- For The First Time, We Have Confirmation That Earth's Core Is Actually Solid □□
- Introducing GitHub Actions
- Ervy - Bring charts to terminal.
- NGINX Unit Now Supports TLS and JavaScript Apps with Node.js - NGINX
- Cloud Academy - Learn serverless full stack development for free
- Setting Up a RESTful API with Node.js and PostgreSQL
- ajv - npm
- fastest-validator - npm
- A Guide to GraphQL in Plain English | SourceCode
- Three ways to represent your GraphQL schema – Apollo GraphQL
- Creating and Reading QR Codes with Node.js |www .thecodebarbarian.com
- A personal review of automated testing tools in the JavaScript world
- How to bring a new tool to your team: 5 tips for successful adoption - CircleCI
- 基于角色和资源的用户权限控制用 SpringMVC实现 ） - u011277123的博客 - CSDN博客
- 每周分享第 30 期 - 阮峰的网络日志
- awk 入门教程 - 阮峰的网络日志
- ES6 In Depth: Arrow functions - Mozilla Hacks - the W eb developer blog
- Microservice Architecture at Medium – Medium Engineering

- JavaScript: What Are Pure Functions And Why Use Them?
- git - the simple guide - no deep shit!
- Gitflow Workflow |Atlassian Git T utorial
- Faster async functions and promises ·V8
- October 21 post-incident analysis |The GitHub Blog
- Node.js Everywhere with Environment V ariables! –Node.js Collection – Medium
- TCP 协议的堵塞控制算法
- Intro to TCP Congestion Control
- Top 10 GitHub Best Practices - datree
- Best Code Components for Node.js - datree □□
- event-stream vulnerability explained - Zach Schneider □□
- Transpiling And Publishing ES9 NPM Modules With Babel 7
- Testing HTTP Requests in Node.js Using Nock -Alligator .io
- Writing Memory Efficient Software Applications in Node.js □□
- Testing your API with Dredd –Ministry of Programming –Medium □□
- 你不知道的 Node.js 性能优化 □□
- Understanding JavaScript's async await □□
- 数据库连接池到底应该设多大这篇文章可能会颠覆你的认知 - 后端 - 掘金

十二月

- 理解递归 □□
- A cartoon intro to DNS over HTTPS
- WebSockets - A Conceptual Deep-Dive
- Copying objects in Javascript
- 这一次彻底弄懂 JavaScript 执行机制 - □□
- Event Loop 必知必会六道题 - 知乎
- 不要混淆nodejs 和浏览器中的event loop - □□
- 19 ways to become a better Node.JS developer in 2019 □□
- Debug Your Node.js App in 60 Seconds –Node.js Collection –Medium
- Node.js API and W eb Frameworks for 2019 |Checkly □□
- InfoQ - GitHub 2018 年十大新开源项目揭晓
- 27道Redis精选面试题会做几题
- Common AsyncA wait Design Patterns in Node.js
- PromisesA+ □□
- Promise 必知必会十道题 - 知乎
- 深入 Promise(一)Promise 实现详解 - 知乎
- How does MySQL Replication really work?
- npm package permissions—an idea
- ECMAScript modules in Node.js: the new plan
- AVA 1.0 □ – □ Sindre Sorhus' blog
- NPM入门 - 基础使用 - kelsen - 博客园

2019 年阅读书单

2019年马就要过去了， 所以就总结了一下这一年中看完的一些书籍

1 月

- 《图解HTTP》
- 《许三观卖血记》
- 《第七天》
- 《兄弟》
- 《大话数据结构》
- 《半小时漫画历史系列四册》

2 月

- 《秋叶： 如何高效读懂一本书》
- 《Linux命令行与shell脚本编程大全第三版》
- 《The Way To Go》
- 《Go Web Programming》
- 《了不起的盖茨比》
- 《白夜行》

3 月

- 《在细雨中呼喊》
- 《程序是怎样跑起来的》
- 《Go In Action》
- 《Linux就该这么学》
- 《面试官：我在树上的女人》

4月

- 《build-web-application-with-golang》
- 《玩儿》

5月

- 《美国众神》
- 《网络是怎样连接的》

6月

- 《平凡的世界》

7月

- 《斗罗大陆

8月

- 《流浪地球》

9月

- 《撒哈拉的故事》

11月

- 《明朝那些事儿全集

12月

- 《electron in action》
- 《围城》

2019 年阅读资源链接

2019年一些已读的文章链接

一月

- ES5和ES6中的继承图 □□
- 一张图理解prototype、proto和constructor的三角关系
- nodebestpractices
- How to scale your Node.js server using clustering
- Threads in Node 10.5.0: a practical intro□□
- We're under attack! 23 Node.js security best practices
- project-guidelines/README-zh.md at master · elsewhencode/project-guidelines · GitHub
- How to use Docker for Node.js development □□
- awesome-cheatsheets/node.js at master · LeCoupa/awesome-cheatsheets · GitHub
- An Overview of Buffers in Node.js □□
- 9 Bash Aliases to Make Your Life Easier □□
- The InterPlanetary File System (IPFS) Simply Explained & Illustrated
- MySQL 索引及查询优化总结□□
- Node.js & JavaScript T esting Best Practices □□
- GitHub - toml-lang/toml: Tom's Obvious, Minimal Language
- Async Stack Traces in Node.js 12 □□
- Using worker_threads in Node.js □□
- Trash talk: the Orinoco garbage collector
- 5 common mistakes in every Node.js app □□
- Aliasing module paths in Node JS□□□□
- 数据库表连接的简单解释
- lets-talk-js-documentation
- 每天一个设计模式
- 【翻译】寒冬中的一年半前端跳槽 - 掘金
- 一个前端的2018总结，2019展望 | 掘金年度征文 - 掘金
- 完美二叉树、完全二叉树、完满二叉树
- 概率计算(抽奖活动、命中率)
- The Twelve-Factor App□□
- How to start a Node.js project □□
- FizzBuzz in 10 languages!
- Creating a Node gRPC Service Using Mali□□
- What is HTTP/2 All About? □□
- Adding tracing with Jaeger to an express application |Rhonabwy
- 不要再问我跨域的问题 - 个人文章 - SegmentFault 思否
- The AIM-lang project series
- mkcert: valid HTTPS certificates for localhost
- Discussing Docker. Pros and Cons.
- Philipp Hauer's Blog

- A Better Way to Develop Node.js with Docker □□
- pi-awesome-code-validation
- Node Weekly 272
- Node Weekly 273
- Build RESTful API service in golang using gin-gonic framework

二月

- 减小docker镜像体积
- why-is-serverless-important
- using-buffers-in-node-js
- build-a-rest-service-with-fastify
- 图解浏览器的基本工作原理
- 图解 HTTP 缓存
- CI/CD - Serverless Ebook using Gitbook CLI, Github Pages, Github Actions CI/CD, and Calibre
- what-every-developer-should-know-about-tcp
- df du
- Grouping and aggregating data using SQL
- 译-保持Node.js 的速度-创建高性能 Node.js Servers 的工具、技术和提示
- 5-git-workflows-to-improve-development

三月

- 有史以来最强的 5G 入门科普！□□
- Docker Tutorial Series – Romin Irani’s Blog□□
- Docker Commands – The Ultimate Cheat Sheet – Hacker Noon
- 容器Docker详解
- How To Build a Node.js Application with Docker |DigitalOcean
- Git Server Deployment
- tmux 入门

四月

- Unicode Character Set and UTF-8, UTF-16, UTF-32 Encoding
- OAuth 2.0 的一个简单解释
- OAuth 2.0 的四种方式
- GitHub OAuth 示例教程
- OPS 入门教程

五月

- Tips to use VSCode more efficiently
- Node.js v12 - New Features You Shouldn't Miss
- Node Best Pratice
- Top Node.js Metrics to Monitor

- A Guide to Node.js Logging

六月

- A guide to setting up Vim for JavaScript development
- 暗网
- Github: fromcodertoexpert

七月

- Node Bigint
- Node Cluster
- good-practices-for-high-performance-and-scalable-node-js-app

八月

- 一致性hash原理
- bridging-nodejs-and-python-with-pynode
- promise.allsettled
- date-fns
- lerna 多包管理开发

十月

- cli增强命令
- js 新特性

十一月

- nodejs-k8s-guide
- deploying-a-node-app-to-google-cloud-with-k8s
- tips-to-write-better-conditionals-in-javascript
- working with github actions
- 7 methods for working with directories in nodejs
- async-generator-functions-in-javascript
- introducing-github-actions

十二月

- 20-ways-to-become-a-better-node-js-developer-in-2020
- lerna
- initial-server-setup-with-centos-7
- 什么是homebrew
- ellwk blog

- electron-tutorial
- what is deno
- 7 lib to build node CLI
- Yarn Workspaces: Organize Your Project's Codebase Like A Pro
- A Practical Guide to Symbols in JavaScript
- Go Introductory Tutorials - rungo

2019年阅读书单

2019年马就要过去了， 所以就总结了一下这一年中看完的一些书籍

一月

- 《人生只有一次去做你想做的事》
- 《骆驼子》
- 《Go Web Programing》
- 《挪威的森林》

二月

- 《挪威的森林》

四月

- 《云边有个小卖部》
- 《你不知道的JavaScript》(上卷)
- 《ECMAScript 6 入门》

五月

- 《钢铁是怎样练成的》
- 《数据结构与算法JavaScript描述》
- 《漫画算法-小灰的算法之旅》
- 《我与地坛》

十月

- 《三体》
- 《Javascript 设计模式与开发实践》

十一月

- 《Redis入门指南》
- 《白鹿》
- 《Redis设计与实现》

十二月

- 《Linux命令行与shell脚本编程大全第三版》

- 《精通Linux第二版》

你不知道的Homebrew

> Homebrew is a package manager for Mac OS. It lets you download binaries, packages, and applications with a single command.

安装Homebrew

```
# Installs Homebrew
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/insta
```

升级Homebrew

```
# Updates Homebrew
brew update
```

搜索包

```
brew seach tree
```

安装包

```
# Install Package
brew install <formula>

# For Example
brew install tree
```

升级包

```
# Upgrades one package to the latest version
brew upgrade <formula>

# Upgrades all packages to their latest version
brew upgrade

# For Example
brew upgrade tree
```

删除包

```
brew uninstall <formula>

# For Example
brew uninstall tree
```

清理空间

```
brew cleanup
```

Brew Tap

`homebrew/core` 维护一份可用包列表在你执行 `brew install` 命令时都是从这个列表进行安装的。

如果需安装是一个第三方的包列表就需我们使用 `brew tap` 命令来将此列表添加新 `homebrew/core` 列表中

```
# For Example
# Required to install MongoDB with Homebrew
brew tap mongodb/brew

# Installs MongoDB
brew install mongodb-community
```

Brew Cask

`homebrew/cask` 可以安装GUI程序， 如： Chrome， Vscode, Atom等

```
# 获取brew cask
brew tap caskroom/cask
brew install brew-cask

# Installs Google Chrome
brew cask install google-chrome
brew cask install visual-studio-code
```

撤销Commit

做笔记做笔记， 做笔记。

撤销commit

通常我写完代码之后就会执行以下命令：

```
git add -A //添加所有文件

git commit -m "注解信息"
```

执行完成之后突然发现有的文件的修改不应该在这次commit提交所以想要撤回commit。之前我的做法是使用sourcetree 来完成。 用着用着感觉好麻烦所以就好好看了下 `git reset` 命令。

其实我们可以这么做：

```
git reset --soft HEAD^
```

上面命令中的 `HEAD^` 的意思是上一个版本也可以写成 `HEAD~1` ； `HEAD^^` 就上上次的版本可以使用 `HEAD~2` ，依次类推即可。所以如果你进行了 `N(N >= 1)` 次commit回滚可以使用 `HEAD~N`

参数解释

- `--mixed`

  不删除工作空间改动代码撤销 commit并且撤销 git add . 操作

  这个为默认参数 ，`git reset --mixed HEAD^` 和 `git reset HEAD^` 效果是一样的。

- `--soft`

  不删除工作空间改动代码撤销 commit不撤销 `git add .`

- `--hard`

  删除工作空间改动代码撤销 commit撤销 `git add .`

  注意完成这个操作后就恢复到了上一次的 commit状态。

修改commit的注释

如果commit注释写错了只是想改一下注释只需

```
git commit --amend
```

此时会进入默认 vim编辑器修改注释完毕后保存就好了。

添加漏加的改动

```
git add file

git commit --amend --no-edit
```

`--no-edit` 参数可以让我们不用进入vim编辑器， 使用原有信息直接修改。

撤销本地的所有更改

撤销本地的所有更改平时我都是通过　vscode编辑器自带的git可视化工具来处理的略嫌**麻烦**。

```
git reset HEAD .  # 撤销暂存区的修改。注意命令最后面一个点

git checkout .  && git clean -xdf  # 撤销工作区的所有更改(包括新增的文件和目录)。注意命

### 或者使用：
git add -A && git reset --hard HEAD
```

Vim 快速入门

Vim是一个类似于Vi的著名的功能强大、高度可定制的文本编辑器在 Vi的基础上改进和增加了很多特性。VIM是自由软件。本篇文章是作者的学习笔记发布本篇文章主要为了备忘同时也希望能帮助读者快速入门 Vim。

新建文件

```
# 方法一
# 直接输入以下命令编写文件内容； ESC + : + wq! filename 保存退出
vim

# 方法二
# ESC + : + wq 保存并退出
vim filename

vim + filename # 打开文件光标定位最后一行
```

保存文件并退回终端

命令模式下双击Z (Z 大写的)

移动

```
# 每次移动一个字符
k 上移；
j 下移；
h 左移；
l 右移；
```

ctrl+f 向前移动一页（page down），ctrl+b 向后移动一页（page up） 更大范围的移动

```
*               当光标停留在一个单词上， * 会在文件内搜索该单词,并跳转到下一处;
#               当光标停留在一个单词上，# 会在文件内搜索该单词,并跳转到上一处；
(/)             移动到 前/后 句 的开始；
{/}             跳转到 当前/下一个 段落 的开始。
g_              到本行最后一个不是 blank 字符的位置。
fa              到下一个为 a 的字符处你也可以 fs到下一个为s的字符。
t,              到逗号前的第一个字符。逗号可以变成其它字符。
3fa             在当前行查找第三个出现的 a。
gg              将光标定位到文件第一行起始位置；
G               将光标定位到文件最后一行起始位置；
NG或Ngg         将光标定位到第 N 行的起始位置
```

快速移动光标

```
H               将光标移动到屏幕上的起始行或最上行  };
M               将光标移动到屏幕中间；
L               将光标移动到屏幕的最后一行。
```

同样需注意字母的大小写。 `H` 和 `L` 命令还可以加数字。如 `2H` 表示将光标移到屏幕的第 `2` 行，`3L` 表示将光标移到屏幕的倒数第 `3` 行。

行内移动光标

```
w               右移光标到下一个字的开头；
e               右移光标到一个字的结尾；
b               左移光标到前一个字的开头；
0               左移光标到本行的开始；
$               右移光标到本行末尾；
^               移动光标到本行的第一个非空字符。
```

搜索匹配

```
/str            正向搜索字符串str；
n               继续搜索定位到  str字符串下一次出现的位置；
N               继续搜索定位到  str字符串上一次出现的位置；
？str            反向搜索字符串str；
```

替换和删除

Vim常规的删除命令是d、x 前者删除 `行` ,后者删除 `字符`

```
rc              用c替换光标所指向的字符
nrc             用c替换光标所指向的前n个字符
x               删除光标所指向的当前字符
nx              删除光标所指向的前  n个字符
dw              删除光标右侧的字
ndw             删除光标邮政的  n个字
db              删除光标左侧的字
ndb             删除光标左侧的  n个字
dd              删除光标所在行并去除空隙
ndd             删除n行内容并去除空隙

d$              从当前光标位置删除字符直到行的结束
d0              从当前光标位置删除字符直到行的开始
J               删除行的回车符（  CR并和下一行合并
```

Vim常规的替换命令有 `c` 和 `s`

```
s               删除光标所指向的字符进入编辑模式
S               删除当前行进入编辑模式
c$              删除光标位置到行尾的所有字符并进入编辑模式
c0              删除光标位置到行开始位置的所有字符并进入编辑模式
```

复制粘贴

从正文中删除的内容如字符、字或行并没有真正丢失而是被剪切并复制到了一个内存缓冲区中。用户可将其粘贴到正文中的指定位置

```
p                   小写字母p,将缓冲区的内容粘贴到光标的后面
P                   大写字母P,将缓冲区的内容粘贴到光标的前面
```

如果缓冲区的内容是字符或字直接粘贴在光标的前面或后面如果缓冲区的内容为整行正文执行上述粘贴命令将会粘贴在当前光标所在行的上一行或下一行。 注意上述两个命令中字母的大小写。Vim 编辑器经常以一对大、小写字母如 p 和 P来提供一对相似的功能。通常小写命令在光标的后面或行操作太写命令在光标的前面或行操作。

复制正文

```
yy                  复制当前行内容到缓冲区
nyy                 复制n行内容到缓冲区
“+y                 复制当前行的内容到操作系统的粘贴板
”+nyy               复制n行内容到操作系统的粘贴板
```

撤销和重复

```
u                   小写字母u,撤销前一条命令的结果
.                   重复最后一条修改正文的命令
```

进入插入模式

在正确定位光标之后可以使用以下命令进入插入模式：

```
i                   在光标左侧插入正文
a                   在光标右侧插入正文
o                   在光标所在行的下一行新增一行
O                   在光标所在行的上一行新增一行
I                   大写，在光标所在行的开头插入
A                   大写，在光标所在行的末尾插入
```

按 `ESC` 键 和 组合键 `Ctrl + [` 可退出插入模式。

打开、保存、退出

```
:e path_to_file/filename      在已经启动的Vim中打开一个文件
:w                            保存当前编辑的文件
:w  new_fileName              将当前文件另存为new_fileName
ZZ                            大写保存并退出
:q                            在未作任何修改的情况下退出
:q!                           放弃所有修改并退出
:wq                           保存并退出注意命令顺
```

行号与文件

```
:n                            将光标移动到指定行同  ngg和nG相同
```

命令模式下可以规定命令操作的行号范围。数值用来指定绝对行号字符 `.` 表示光标所在行的行号字符符 $表示正文最后一行的行号简单的表达式例如 `.+5` 表示当前行往下5行,例如:

```
:345                          将光标移动到第345行
:345w file                    将第345行内容写入file文件
:3,5w file                    将第3至第5行写入file文件
:1,.w file                    将第1行至当前行写入file文件
:.,$w file                    将当前行至最后一行写入file文件
:a,bW file                    将第a行至第b行的内容写入file文件
:r file                       读取file文件的内容插入当前光标所在行的后面
:e file                       编辑新文件file 代替原有内容
:f file                       将当前文件重命名为file
:f                            打印当前文件的名称、状态、行数、光标所在行号等
```

字符串搜索

在 编辑模式 讲过字符串的搜索此处的 命令模式 也可以进行字符串搜索给出一个字符串可以通过搜索该字符串到达指定行。如果希望进行正向搜索将待搜索的字符串置于两个 `/` 之间如果希望反向搜索则将字符串放在两个 `?` 之间

```
:/str/                        正向搜索将光标移动到下一个包含字符串  str 的行
:?str?                        反向搜索将光标移动到上一个包含字符串  str 的行
:/str/w file                  正向搜索将第一个包含字符串的行写入  file 文件
:/str1/,/str2/w file          正向搜索将包含字符串  str1 的行至 包含字符串 str2 的行的内

:d                            删除光标所在行
:nd                           删除 n 行

:recover                      恢复意外退出而没有保存的修改也可在启动的时候使用 '-r'选项
```

Shell 切换

当处于编辑的对话过程中时可能需执行一些 Linux命令。如果需保存当前的结果退出编辑程序再执行所需 Linux命令然后再回头继续编辑过程就显得十分累赘。如果能在编辑的环境中运行Linux命令就要省事得多。在Vim中可以用下面命令来做到这一点：

```
:!shell_command             执行完 shell_command 后回到Vim  :!pwd
```

分屏

```
:split                      缩写:sp, 上下分屏
:vsplit                     缩写:vsp,左右分屏1
```

另外也可以在终端里启动 vim时就开启分屏操作：

```
vim -On file1 file2...      打开 file1 和 file2 垂直分屏
vim -on file1 file2...      打开 file1 和 file2 水平分屏
```

快速操作

```
以下命令可以对标点内的内容进行操作。
ci’、ci”、ci(、ci[、ci{、ci<                分别更改这些配对标点符号中的文本内容
di’、di”、di(或dib、di[、di{或diB、di<       分别删除这些配对标点符号中的文本内容
yi’、yi”、yi(、yi[、yi{、yi<                分别复制这些配对标点符号中的文本内容
vi’、vi”、vi(、vi[、vi{、vi<                分别选中这些配对标点符号中的文本内容
```

删除全部

```
# 光标在第一行
V + G + d

# 光标在任一行
1,$d
```

快速删除

```
# 有时候我们需删除括号或者引号内的内容
# 方法一：
定位到{},()等位置按  v然后按%选中删除

# 方法二:
vi + c          # c 为 " ' ( { [ 等字符
```

替换

```
:[range]s/pattern/string/[c,e,g,i]

range  指的是范围，1,7 指从第一行至第七行，1,$ 指从第一行
       至最后一行也就是整篇文章也可以     % 代表。
pattern  就是要被替替换的字串可以用   regexp 來表示。
string   pattern将由string所取代。
c  confirm每次替换前询问。
e  不显示error。
g  globe不询问整行替换。
i  ignore 不分大小写。
```

vim中大小写转化

```
# 整篇文章转换为小写
ggguG

gg=光标到文件第一个字符
gu=把选定范围全部小写
G=到文件结束

# 整篇文章转大写
gggUG

# 只转换某个单词
guw, gue
gUw, gUe

# 其他
gU0         从光标所在位置到行都变为大写
gU$         从光标所在位置到行尾都变为大写
shift + `   光标所在字符大小写切换  , 配合v可以进行选择转换
```

其他

```
:files 或 :buffers 或 :ls 列出目前buffer中的所有文档

:e filename 进入vim后在不离开  vim的情形下再打开其他文档

:e# 或 Ctrl + ^, 编辑前一个文档，  用于两文档相互编辑时相当好用

:sh 退回终端命令界面以执行命令，     exit回到vim界面

:K 大写的K, 会打开光标所在单词的manpage

:r !command 在光标下一行插入command的输出, 报错时会插入错误信息
```

XSftave Stp

XXX 软件搭建过程你懂的！ 放自我 ~

1. 登录服务器

```
ssh username@ip
```

2. 下载Brook文件

```
# 根据最新版本替换url
wget https://github.com/txthinking/brook/releases/download/v20190205/brook

# 增加可执行权限
chmod +x brook

# 根据配置文件中的command来移动文件
mv brook /usr/local/bin/
```

3. 安装supervisor

```
apt install supervisor
```

4. 编辑supervisor配置文件

B ：修改 `PORT` 和 `PASSWORD` 信息,将其命名为 `ssserver.conf` , 然后将其放到 `/etc/supervisor/conf.d` 目录中即可。

```
; supervisor config file
[program:ssserver]
command=/usr/local/bin/brook ssserver -l 0.0.0.0:PORT -p "PASSWORD"
autostart=true                                                ; start at supervis
startsecs=1                                                   ; # of secs prog mu
startretries=3                                                ; max # of serial s
autorestart=true                                              ; when to restart 
exitcodes=0,2                                                 ; 'expected' exit c
stopsignal=QUIT                                               ; signal used to ki
stopwaitsecs=10                                               ; max num secs to w
stopasgroup=false                                             ; send stop signal
killasgroup=false                                             ; SIGKILL the UNIX
redirect_stderr=true                                          ; redirect proc std
stdout_logfile=/tmp/brook.stdout.log                          ; stdout log path,
stdout_logfile_maxbytes=1MB                                   ; max # logfile byt
stdout_logfile_backups=10                                     ; # of stdout logfi
stdout_capture_maxbytes=1MB                                   ; number of bytes 
stdout_events_enabled=false                                   ; emit events on st
stderr_logfile=/tmp/brook.stderr.log                          ; stderr log path,
stderr_logfile_maxbytes=1MB                                   ; max # logfile byt
stderr_logfile_backups=10                                     ; # of stderr logfi
stderr_capture_maxbytes=1MB                                   ; number of bytes 
stderr_events_enabled=false                                   ; emit events on st
```

5. 重启supervisor

```
service supervisor restart

# 查看ssserver进程是否正常运行
supervisorctl status
# ssserver                         RUNNING   pid 2824, uptime 0:15:28
# RUNNING 表示成功运行了
```

6. 根据上面填写的PORT和PASSWORD填写Shadowsocks服务器配置即可

使用Let Encrypts,HTTP 你的网站 (Ubuntu)

HTTPS全称： Hyper Text Transfer Protocol over Secure Socket Layer 或 Hypertext Transfer Protocol Secure超文本传输安全协议是以安全为目标的 HTTP通道简单讲是 HTTP的安全版

Note: 在申请证书之前 域名配置需在于 nginx 的配置文件中, 且域名需监听 `80` 端口;

1. 安装 cert 命令行工具

```
$ sudo add-apt-repository ppa:certbot/certbot
$ sudo apt-get update
$ # for apacha: sudo apt-get install python-certbot-apacha
$ sudo apt-get install python-certbot-nginx
```

2. 申请证书

```
$ sudo certbot --nginx -d YOUR_DOMAIN
$ sudo certbot --nginx -d YOUR_DOMAIN_A -d YOUR_DOMAIN_B # 申请多个域名证书
```

NOTE: 将 `YOUR_DOMAIN` 、 `YOUR_DOMAIN_A` 、 `YOUR_DOMAIN_A` 替换成真实的域名

3. 证书过期重新申请

Let Encrypts 的证书只只支持 90 天的时间, 如果过期需从新申请 , 执行以下命令:

```
sudo certbot renew --dry-run
```

生活杂记(一)

生活不只是苟且！

> 却也没有诗和远方！

一

岁月如刀啊
两鬓白发满脸的皱纹以及佝偻的身影
那是岁月留下的痕迹
于是心生焦虑奋起直追
浩瀚书海虽是囫囵吞枣只因怕来不及
愿岁月停留
等等迷失途中的孩子

二

从前的恣意妄为是何等的幸福，
如今的无能为力是何等的悲哀，
快快成长吧努力抓住现在所拥有的一切，
把生活过成我们想要的样子。
希望我们都能过的好像照片里那样好！

三

我把你弄丢了
推向他身边
剩我一人在深夜寂寞
还记得那几年
你微笑的脸是那么甜
回想起来一切还是那么美
你现在在哪过的还好吗
一切仿佛都好像在昨天
好想你还在我身边想把你宠的像孩子样天真
可现在你在他身边笑的还是那么甜孩子一样的天真
你那孩子般的天真我看不见
我再也看不见
希望你以后还笑的那么甜

四

我曾经天真的以为用真心对任何人就可以得到真正的友情真正的爱情。
后来认识了一些人经历了一些事才知道一切都是我以为。

我以为你会一直瞧到最后可惜不是你到最后剩下的孤寂我一个人享受就好了即使再痛又有什么关系。

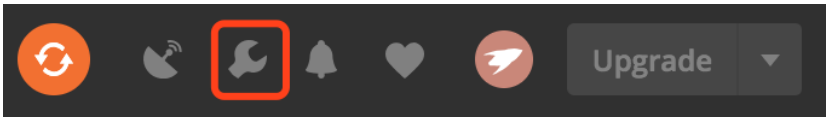
相遇总是猝不及防离别多是蓄谋已久总有一些人会慢慢淡出你的生活你要学会接受而不是怀念。
留不住的心就让它吧。吧！

Postman 使用自签名证书发送https 请求

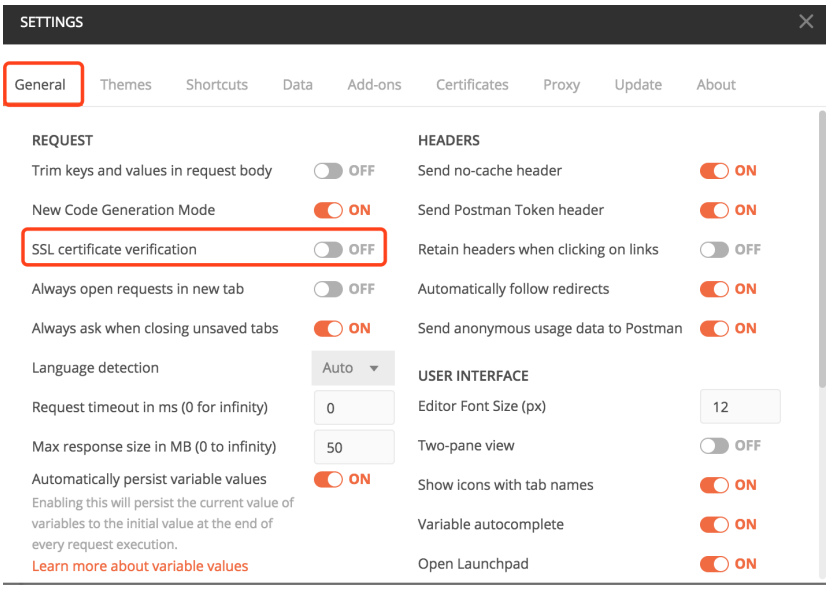
做笔记做笔记， 好记性不如烂笔头。

设置步骤

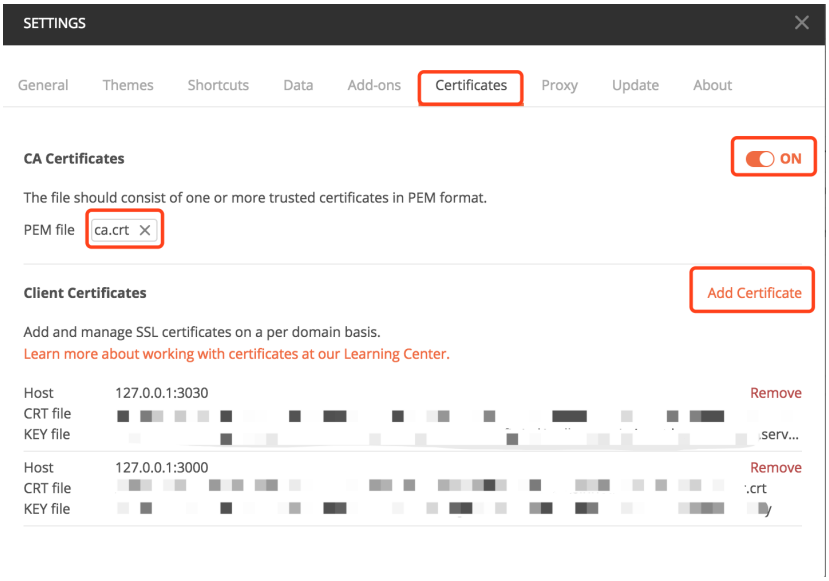
1. `Command + ,` 打开Settings 面板 或点击右上角的**小扳手图标**，如下图：



2. 在General 面板中关闭 SSL certificate verification ，如下图：



3. 切换Certificates 面板
4. 开启CA Certificates ，选择ca.cert文件
5. 点击Add Certificates 按钮
6. 根据提示填写URL地址和端口，选择对应的cert、key文件


SETTINGS
General
Themes
Shortcuts
Data
Add-ons
Certificates
Proxy
Update
About
CA Certificates
ON
The file should consist of one or more trusted certificates in PEM format.
PEM file
ca.crt
Client Certificates
Add Certificate
Add and manage SSL certificates on a per domain basis.
Learn more about working with certificates at our Learning Center.
Host
127.0.0.1:3030
Remove
CRT file
KEY file
Host
127.0.0.1:3000
Remove
CRT file
KEY file

结构化查询语言学习

SQL(Structured Query Language) 是结构化查询语言的简称它最重要的是数据库的操作语言。 对于学习数据库而言学习 SQL的使用是相当的重要!

语句的分类

1. `DQL`： Data Query Language,数据查询语句用于数据的查询
2. `DML`： Data Manipulation Language,数据库操作语言， 主要用于数据的插入、更新、删除。
3. `DDL`： Data Definition Language,即数据定义语言主要用于表的创建、删除修改等

DDL语句

建表语句

```
CREATE TABLE student ( -- student
    id INT SERIAL PRIMARY KEY, -- PRIMARY KEY 代表表的主键是行的唯一标识不能重复。
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    phone varchar(32)
    age INT
);
```

删除表

```
DROP TABLE table_name;
```

创建索引

```
CREATE INDEX idx_student_phone ON student USING gin(phone);
```

DDL语句

插入数据

```
INSERT INTO student VALUES(1, 'Chui', 'Dylan', '131xxxxxxxx', 22);
INSERT INTO student VALUES(2, 'ming', 'xiao', '0123456789', 15);

-- or , 某些省略的字段会被置空
INSERT INTO student (first_name, last_name, phone, age) VALUES ('Chui', 'Dylan
```

更新数据

```
UPDATE student SET age = 21; -- 更新全部数据

-- 根据条件更新多列数据
UPDATE student SET age = 20, phone = '131zzzzzzzz' WHERE id = 1;
```

删除数据

```
DELETE FROM student WHERE id = 2;

-- 删豫表数据
DELETE FROM student;

-- TRUNCATE 语句： 属于DDL, 相当于用重新定义的方法丢去原有的数据，
-- 而DETELE属于DML, 是一条一条数据的删豫以    truncate 执行会更快
TRUNCATE student;
```

DQL语句

```
-- 单表查询
SELECT * FROM student; -- * 代表所有列

-- 查询指定列
SELECT id , first_name, last_name FROM student;

-- select expression
SELECT 2 + 3, id || first_name FROM student WHERE id = 2;

-- 过滤条件
SELECT id , first_name, last_name FROM student WHERE id = 2;

-- 排序
SELECT * FROM student ORDER BY id ASC; -- ASC 代表升序(默排序方式 ), DESC 代表豫

SELECT * FROM student WHERE first_name LIKE '%chui%' ORDER BY id ASC, age DESC;

-- 分组查询
SELECT age, COUNT(*) FROM student GROUP BY age; -- 使用GROUP BY时需使用聚合函数

-- JOIN
SELECT t1.uid, t1.user_name, t2.email, t2.auth_identifier FROM user_info t1, us

SELECT t1.*, t2.* FROM user_info t1 left join user_auth t2 on t1.uid = t2.user_

-- insert into ... select from table_nme(需先创建表 ),快捷备份表的一个方法
INSERT INTO student_bak SELECT * FROM student;

-- 联合
SELECT * FROM student WHERE id = 2 UNION SELECT * FROM student_bak WHERE id = 2

-- UNION 会将两条相同的数据合并如果不想合并使用    UNION ALL即可
SELECT * FROM student WHERE id = 2 UNION ALL SELECT * FROM student_bak WHERE id
```

bntu 初始化设置

当你第一次创建ubuntu 16.04版本的服务器时有一些配置步骤为基础配置的一部分是需你尽早设置的这将增加服务器的安全性和可用性并为后续操作提供坚实的基础。

01. Terminal 远程登录云服务主机

1. 登录阿云服务控制台
2. 面侧找到 `云服务器ECS`，点击找到服务器 `实例`
3. 复制 `公有IP地址`
4. 本地打开 `Terminal`，通过下面命令登录

```
ssh username@ip
```

5. 密码验证， 输入购买时设定的服务器密码如果忘记了可以在实例界面重置密码

02. 管理用户

1. 添加用户

```
$ sudo adduser username  # 然后根据提示输入信息
```

2. 赋予新用户sudo权限

```
$ sudo usermod -aG sudo username

# annother way
# $ visudo
#
# add this line below ' User privilege specification area'
# usernmae  ALL=(ALL:ALL) ALL
# 用户名     所有主机 = 所有用户可执行命令所有组可执行命令）     所有命令
```

3. 修改用户密码

```
$ sudo passwd username
```

4. 删除用户

```
$ sudo deluser username

# 连同用户home目录一起删除
$ sudo deluser --remove-home newuser

# 如果该用户用于sudo权限   通过visudo命令删除应用户对应行记录即可
```

03. SSH Auth 无密码登录（安全可靠）

1. 在本地生成秘钥

```
ssh-keygen       # 不要输入密码

# 默在用户  .ssh目录下。
```

2. 将公钥上传至远程服务器

```
mkdir .ssh
touch  .ssh/authorized_keys
# 手动复制本地公钥内容到`authorized_keys`

# ssh 命令方式推荐）
ssh-copy-id username@host
```

3. 修改文件权限

```
chmod 777 .ssh
chomd 644 .ssh/authorized_keys
```

4. 禁用root用户SSH 登录

```
# 修改 PermitRootLogin no
vim /etc/ssh/sshd_config [etcsshsshdconfig]
```

5. 禁用基于密码的登录

```
# 添加 `PasswordAuthentication no`, 保存退出
sudo vim /etc/ssh/ssh_config

# 重启ssh auth 服务
sudo service ssh restart
```

04. 设置sudo 不需要输入密码

```
sudo vim /etc/sudoers

# 修改对应一行
# %sudo   ALL=(ALL:ALL) NOPASSWD: NOPASSWD: ALL

# 详情请参考 man sudoers
```

05. 安装Docker

1. 参照 docker官网
2. 将 用户添加到 `docker` 组

```
sudo usermod -aG docker username
```

06. 阿里云服务器公网开放端口

1. 登录阿里云服务器控制台
2. 实例操作点击 `更多` 选择 `安全组配置` 进入实例安全组配置界面选择 `配置规则`
3. 点击 `添加安全组规则` ，在弹出界面输入参数
4. 重启服务

07. 参考文章

- How to Add and Delete Users on Ubuntu 16.04
- How To Edit the Sudoers File on Ubuntu and CentOS
- How To Set Up SSH Keys on Ubuntu 16.04

Speeches

A collection of excellent english speeches.

Don't You Quit!

Every champion has felt it. Every victorious person has felt it. The urge to quit. Don't you give up on your dream. I don't care if you don't have the money, if you don't have help, and you don't have the family for it, and if you don't have the friends for it. Don't you give up on your dream. Don't you do it.

每个冠军，每个胜利者，都曾经想要放弃。不要放弃你的梦想！不管你有没有钱，不管有没有人帮助你，不管你的家人是否支持你，不管你的朋友是否支持你，你都不要放弃梦想！千万不要！

It may take you twice as long. You may have to take courses and classes. You might not read as fast. You might not move as quick. You might not have as much. But don't you quit. 可能你需多花很多时间可能你需多参加很多课程，可能你读得没别人快可能你的行动不如别人敏捷可能你拥有的没有别人多但是你一定不要放弃。

If you lay dead even the animals won't bite you. The risk of being bitten is the cost of getting up. And you have to decide. Are you so concerned about being bitten that you're willing to spend the rest of your life lying dead? Or is there something pumping down inside of you that is saying bite me or not, I'm getting up.

如果你混吃等死连野兽都不屑于啃你一口。如果你害怕被吃掉那么你就永远爬不起来。你要自己做决定你真的那么害怕被吃掉吗害怕到你宁愿混吃等死也不愿意起来奋斗吗又或者其实你身上有一股冲劲不管会不会被吃掉我都要站起来！

I'm going to be the best me. I'm gonna do all that I can do. I'm going for it. Anybody can do anything they set their mind to. But it depends on how bad you want it. With enough persistence most things that seem impossible become possible. You just have to show up or show at so often. Time after time you gotta wake up and you got a drive. You got to be driven to get something done now.

我要做最好的自己我要付出所有的努力我要朝着目标去奋斗。有志者事竟成。但是这取决于你的愿望有多强烈。只要付出坚持不懈的努力一切皆有可能。你要让世人看见你的努力你要一次次地觉醒你要找到自己的动力你要让一股力量驱使你去实现梦想。

You got to be willing to stand on your beliefs. Cuza lot of times you're the only one who can see it. And other people don't see it really. But you gotta really believe in what you've got inside of you that's telling you and driving you to a certain goal. Every minute you decide upon something you know that's what you want. You know you're going to do it. All of these negatives that have been bothering you. They pick up their baggage and get out. They can't live any positive mind. You have to move with courage, with faith, with determination.

你要勇于坚持自己的信念。因为很多时候和其他人相比你才是最清醒的人。但是你必须自己的梦想有信心要对你的目标和动力有信心。每次做决定你都清楚地知道这就是你想要的你知道自己要努力实现它。那些曾经让你心烦意

乱的消极的声都被清理得一干二净因为积极的心态容不得它们存在。你要勇往直前你要心怀信念你要意志坚定。

I don't care what it is. I don't care how difficult it is. Listen, you better put some strength in your back. Plant your feet and stand up. Don't you dare sit down. Don't you dare crumble under the weight of it. Don't you dare give in because of the nature. Don't you dare go home and raise the white flag of surrender because whatever is over you defines you and whatever defines you limits you.

我不管你的梦想是什么我不管它有多难实现。听着你必须自己力量。稳住脚跟站起来不要再坐下去了不要再让压力把你压垮了不要因为自己的惰性而屈服不要自己认输然后灰溜溜地躲回家去能承受多大的压力决定了你是什么样的人。而你是什么样的人决定了你能成多大的事。

This is not your season to die. This is not your season to quit. There's joy. There's peace. There's breakthrough. There's provision. But you'll never see it if you don't stand up to it. Your pain is going to be a part of your prize, a part of your product. I challenge you to never give up. Never give in. And finally, guys, you gotta want to succeed as bad as you want to breathe.

不要因此一蹶不振不要因此半途而废。无尽的快乐内心的平静大的突破，大好的前途就在前方。但是如果你不勇敢面对现实你就永远看不到这些美好的东西。你的痛苦将会成为你的荣誉你的成就。你敢不敢坚持下去永不言弃朋友们你要渴望成功成功和你的呼吸一样重要。

That's just kind of the way that life works sometimes. It's Murphy's Law. When things go wrong, they always seem to happen at once and they just compound on top of each other and it's, it's pretty easy sometimes to feel beaten up. When you're faced with all of those issues and all those problems and they all hit you at the same time that doesn't mean give up. In fact, it means the opposite. It means it's time for you to fight harder to dig in. It means it's time for you to go on the warpath.

有时候生活就是这样的这就是墨菲定律。一事不顺事事不顺很容易让你感到沮丧。当你同时面临很多困难多问题不能放弃。事实上你更加应该坚持下去。因为这些事情意味着你要加倍努力你要埋头苦干。这意味着你要走上战场去勇敢地搏斗！

Look For The Good In Your Life

Whatever you focus on you will find. If you search for negativity in this world, you will find plenty of it. If you search for hate, anger, violence and sadness you will find it.

你关注什么你就会收获什么。如果你寻找负能量你就会发现很多负能量如果你寻找憎恨、愤怒、暴力和悲伤你也会找到。

But the same is true on the flip side: If your only intention is to search for the good, you will find only the good. Whatever meaning you give your life, becomes your life.

但是另外一面是正确的。如果你的目标是寻找美好那么你只会找到美好。你赋予你的生命什么你的生命就会成为什么。

It can be a failure or a lesson. Heart break or character building. Life is against you, or making you stronger.

失败还是教训伤心还是塑造品格生活是故意刁难还是让你变得强大全看你自己的态度。

Because there is no such thing as reality. We choose our own reality by the meaning we give each moment in our lives. Make it your intention to look for the good in your life. To notice the good in others. To be grateful for what you do have. To see challenges as opportunities to show your true character.

因为根本没有“现实”这回事全凭我们自己的视角。所以一定要寻找美好的东西珍惜自己所拥有的。把挑战当作是锻炼自己的机会。

Remember: What you give your attention to will become your experience in life. Practice seeing the good in your life, and in others. Think the best, expect the best and always ask yourself how can this benefit my life. 记住你的关注点在哪里你的成长就在哪里。多关注生命中美好的人和事思考最好的期待最好的，而且时刻问自己这件事怎么能让我的生命受益？

Leave who you were. Love who you are and look forward to who you will become.

接纳过去的自己爱现在的自己期待将来的自己。

Make Excuses or Make Changes

What happens with students, and I see this all the time. They start university but with the mindset that they're constantly looking for excuses, about why things aren't going the way they want, why they're disappointed with their exam results, why they don't think their professor is teaching them effectively, why they keep getting up late and missing classes.

学生的情况我见得多了。他们上了大学但他们的心态不健康。如果发现事情不顺或者对考试结果感到失望或者觉得教授没有认真上课或者一直睡懒觉、旷课他们就会找借口。

And when you do that, you take all the power you have in your life, and you give it away. You just throw it all away, because you're blaming other people. By blaming your professors, or your university, or your friends or family or your circumstances, you give your power away to them, but you need to embrace it.

如果你总是找借口其实就是在放弃自己的主动权把决定权交给别人。因为你总在抱怨别人所以其实你抛弃了自己的主动权。如果你总是发牢骚抱怨老师埋怨学校埋怨朋友埋怨家人埋怨现状那么你就把主动权交给了他们。然而，你应该掌握自己的主动权。

Think about the lies that you tell yourself to rationalize being lazy. When you take the easy road and you leave discipline behind, when you say to yourself, I'll start studying for my exam tomorrow, and then tomorrow comes and you say, I'll start studying for my exam tomorrow. And the exam deadline is getting closer and closer, and a few days before the exam you realize you haven't started studying yet. And the panic sets in and the sleepless nights begin. It's too late.

想一下你曾经说了多少谎言自欺欺人掩饰自己的懒惰。你选择安逸放弃自律。你对自己说明天我就要开始复习考试第二天又继续说我明天我就要开始复习考试。考试一天天逼近开考前几天你终于意识到自己完全还没开始复习。你会感到恐慌睡觉都睡不安稳。但是一切都已经太迟了。

It's the day of the exam you sit at your desk. You open the exam paper for the first time and you experience that all-too-common sinking feeling when you realize you don't know the answer to any of the questions. You know what I'm talking about. We've all been there. It's the excuses and the lack of discipline that are taking you down paths that you shouldn't be going.

考试当天你坐在桌子旁第一次打开试卷发现自己不知道怎么解答接着感到心一沉这种感觉非常普遍。你很清楚我在说什么因为我们都经历过这样的情况。找借口不自律这让你误入歧途。

There are a million reasons why you're disappointed, but the reasons are not important. What is important is what you do about it. What is important is how you're going to spin it so that you take advantage of the situation. Failed an exam? Good. Study two hours a day extra from now on. Had an argument with a friend? Good. You just had a crucial lesson in social science. Try to stay more

calm next time. Keep waking up late and getting to class late? Good. Go to sleep earlier and wake up earlier tomorrow. Whatever your problems are, they are fixable. You just have to persevere.

也许有千百种事情让你沮丧然而它们并不重要重要的是你如何应对重要的是你怎么扭转局势怎么利用现有条件。考试失败了没事从今天开始每天加量学习两小时。和朋友吵架了没事你得到了社交中很宝贵的教训下次记得要更加冷静总是睡懒觉、上课迟到没事早点睡明天早点起。无论你有什么问题可以解决关键是你要能坚持。

And as you work at it brick by brick, you might start out awfully inefficient and incompetent at studying. But being willing to put in the work, and grind it out, and becoming someone who people actually look up to and admire, at that moment in your life, you'll realize it was worth it. All the pain, the suffering, the late-night study grinds. It was all worth it.

你付出点点滴滴的努力也许一开始的时候你学得不太好但是只要你愿意付出愿意用功最后人们都会佩服你欣赏你。到那时你就会知道一切都是值得的。曾经的痛苦磨炼无数个努力学习的不眠之夜都是值得的。

Pain Resolute Adapting

You've probably heard this before that tough times don't last forever but tough people do. That held true when you first heard it and it still holds true today. There will come a time in your life where you may cry and feel like your soul is dying. Those days when you struggle, want to give up, and don't want to face the world. These are the days that define you. Don't quit.

你可能听说过艰难时期总会过去坚强的人才能笑到最后。这个道理古今通用。在人生中你可能会遇到艰难期。你哭泣你感到自己的灵魂在死去。这种时候你痛苦地挣扎你想放弃你不想面对这个世界。然而这正是磨炼你的时候。不要放弃。

Difficulties come to make you strong, but that's only when you make a decision to go on, and you stand by it. Abraham Lincoln said, I'm not concerned that you have fallen and concerned that you will rise. Giving up as human nature brought about by fear, fear of, what if I fail and try, and then fail again. Wouldn't it just be easier to just quit? But here is what is wrong with that. When you give in to that fear, you sentence yourself to a lifetime of failure, a lifetime of regret, a situation where you lose faith in yourself and in humanity. Don't do that to yourself.

只有在你下定决定要坚持到底的时候困难能让你更加强大。亚伯拉罕·林肯曾经说过我不在乎你有没有跌倒我在乎的是你有没有站起来。放弃是人类的天性，它是由恐惧引起的。要是我失败之后再次尝试又失败了怎么办直接放弃这不是更简单吗但是这种想法是不对的。当你向恐惧低头的时候你注定了一生都要失败一生都要后悔你会对自己失去信心对人性失去信心。不要这样对待自己。

You must understand that difficult times will come. There's absolutely nothing you can do about that. What you can do, however, is decide what you're going to do with the times. Life is 10% what happens to you and 90% how you react to it, so you must decide to make a commitment to face it and deal with it, because at the end of pain is success.

你必须明白艰难期总是会来的你无法阻止它。然而你能做的就是决定自己要如何面对艰难期。生活中重要的不是发生了什么而是你如何应对。所以你必须决定好下定决心去面对困难理解困难为痛苦的尽头就是成功。

Name what it is that is holding you back. It may be fear. It may be procrastination, a relationship, self-doubt, or even depression. Whatever it is, identify it. You know you can, if you think deeply about it. Now write it down. This will put into perspective what you have to deal with. The big picture of your life will begin to become clear when you start paying attention to the tiniest details. Start with identifying what is holding you back.

找出到底是什么东西拖了你的后腿可能是恐惧可能是拖延可能是一段关系，可能是系我怀疑甚至可能是抑郁。无论它是什么都要找到它。如果你认真思考你就能找到它。然后把这个东西写下来。这会让你清楚地看到你到底要处

理什么。当你开始注意人生中最小的细节时你人生的大局就会变得清晰起来。找到拖你后腿的东西从这一步做起。

Next, you have to let go of your anger and hate from the past. This is to help you heal yourself. You cannot be happy and productive if you hold on to old hurts. Let go of relationships and situations that are out of your control. Don't try to force it. You'll only end up hurting yourself some more. Your mental health is important, as it guides every other aspect of your life, your physical life, your spiritual life. They all become better when you maintain a healthy mental state. Let go of your past hurt.

接下来你要放下过去的愤怒和怨恨这能帮助你治愈自己。如果你总是沉溺于过去的伤痛那么你不可能快乐也不可能有所成就。放下那些你无法控制的人际关系和情况不要勉强了。如果你放不下你只会进一步伤害自己。你的心理健康是很重要的因为它引导着你生活的每一个方面的生理健康你的精神生活。如果你能保持健康的心理状态你生活的方方面面会变得更好。放下过去的伤痛吧。

Rock Bottom

The thing about life is it's not always easy, and you aren't always winning. Sometimes it hits you at the point of your life, it hits you hard, it hits you really really hard.

生活的真相就是生活很难，你不可能一直是赢家。有时候生活会打击你，非常非常沉重的打击。

The person you love doesn't love you back, you get fired, you lose a family member. And at the point of your life, you're gonna hit rock bottom.

你爱的人不爱你你被辞退了你失去了亲人。有时候你会跌入低谷。

You're paralyzed, you're like, "Why?" And that why can really really destroy you. Once you start asking yourself, "Why me?" "Why not the others," "why me?"

你已经麻木了你会问为什么。“为什么”会摧毁你。一旦你开始问你自己，“为什么是我”为什么不是别人”为什么是我？

I'm actually a good person, I never did something significantly bad, why the hell did it hit me? Because that's life. Life is unfair.

我真的是个好人我从来没有做过严重的坏事但为什么偏偏是我遇到这么多坎坷？”因为这就是生活生活就是不公平的。

Success is not based on the days when the sun shines. Success is based on the dark, stormy, cloudy days. And if you can't absorb failure, you're never gonna get success.

成功不是用阳光灿烂的日子来衡量的成功是由那些黑暗的、狂暴雨、阴霾遮蔽日的日子来衡量的。如果你无法接受失败你就永远不会成功。

Sometimes it takes things falling apart, for better things to fall into place. Sometimes it takes the most uncomfortable path, to lead your life to the most beautiful place.

有时它会让事情分崩离析让更好的事情发生不破不立。）有时候要走最不起眼的路才能把你的生活引向最美丽的地方。

There's gonna be bad days, there's gonna be dark days, but you've gotta embrace it, Because that pain is what makes you stronger. Failure is what makes you stronger.

会有不好的日子会有黑暗的日子但你必须拥抱它因为痛苦会让你更坚强。失败使你更坚强。

You have to accept those downturns, because once you realize those downturns, are just as much part of life as anything else, you're able to strive again.

你必须接受那些低潮时期因为一旦你意识到那些低潮时期和其他任何事情一样是生活的一部分你就能够再次奋斗。

You ll never see the purpose of the storm, until you see the grow it produced. You ll never understand why you went through what you went through, until you see the strength, the power, the resilience that it built inside of you

直到你看到暴风雨成你才明白他的目的。你永远不会明白你为什么要经历你所经历的一切直到你在内心建立起毅力、力量和韧你才能明白你为什么你要经历你所经历的一切。

Ask yourself why. But this why is a better why, a winding this? Why am failing? Why am even getting yourself in a situation where i could fail? Because i have a dream. Because i have goals."

问问你自己为什么。但这就是为什么"我为什么要这么做为什么我会失败为什么我会让自己陷失败的境地因为我有一个梦想。因为我有目标。"

And the more you're thinking back to those original goals, the easier it is for you to get back up and say, "Alright", it might be difficult, it might be painful, it might be stressful, there might be no people that believe in me, but i believe in myself.

你越是想回到最初的目标你就越容易站起来说"好吧"这可能是困难的可能是痛苦的可能是压力的可能没有人相信我但我相信我自己。

You know it might have been the case, that you should have gone through that hardship, break up that you should have gone through that heavy loss, just in order to find something even better, but the only way to get to that even better, is to get back up and back.

你知道可能是这样的你应该经历那种严酷的粉碎的你应该经历那种沉重的损失只是为了找到更好的东西但唯一能找到更好东西的方法 — 就是重新站起来努力奋斗。

To get back up and pick yourself out there again, Arise for that again, stronger, better, smarter, ready to grasp that new opportunity.

重新振作起来重新振作起来重新振作起来变得更坚强更好更聪明准备抓住新的机遇。

You gotta believe the tables in your life will turn, that pain will become power, that weakness will become strength, and that confusion will become peace, better things are coming for your life.

你要相信你生命中的桌子会转动痛苦会变成力量软弱会变成坚强混乱会变成和谐更好的事情会降临到你的生命中。

Every day is a new beginning. It's time for you to start treating it that way.

每一天都是新的一天该是时候认真对待它了。

Some risks you should definitely take in life

Risks are a part of all our lives. Some you take, some you don't. Actually, any decision you make has an element of risk. Nothing comes with a full warranty. Whenever you attempt to do anything, failure is a possibility. However, there are certain risks worth taking that can contribute significantly to your happiness. So, here are some risks you should definitely take in life!

风险是人生的一部分。有些风险你会去承担的有些则不然。实际上你做的每个决定都包含了风险有什么事情是保证万无一失的。每当你尝试做一件事你都可能会失败。然而有些风险值得你去承担的它们能让你变得更快乐。那么接下来将要介绍给你的就是一些人生中你应该承担的风险

Number 1 - Pursuing Your Dreams. Sure, many people have followed their dreams with passion, only to experience complete failure. What's even more tragic, however, is that most people never risk really pursuing their dreams. Everyone should make a serious attempt to realize their dreams at least once in life. After all, you don't want to find yourself sitting somewhere, old and decrepit, and have to ask yourself. What if I had done it? What kind of life could I have had?

第一追寻你的梦想。确实很多人满怀热情地追寻梦想到头来却彻底失败了。然而更悲哀的是有些人从来不敢追求自己的梦想。在人生中每个人都应该认真地尝试实现自己的梦想至少要有一次这样的经历。毕竟你可不想在白发苍苍时才回头问自己如果当初我实现了梦想呢我的人生会是什么样的？

Number 2 - Risk Rejection. The reason we get rejected usually has much more to do with the person who turned us down than with us. Regardless of what's at stake – whether you ask someone out on a date or just want a bit of advice – there will be some people that just won't comply with your wishes. The fear of rejection is what keeps many people from asking for what they want. It often seems better not to ask at all than risk a no. Yet, if you don't even ask, how can you expect to get a yes.

第二承担被拒绝的风险我们被拒绝更多地是因为拒绝我们那个人而不是因为我们自己有问题不管是什么事比如你想与某人约会或者想得到一点建议，总会有人不如你愿。很多人不敢去索要自己想要的东西因为他们害怕被拒绝。貌似与其被拒绝还不如不要开口问。然而如果你连问都不敢问那你就别指望人家答应你了。

Number 3 - Experiencing Something New. When you learn or try something new, there's always an element of risk. Skiing, sky-diving - even learning how to skate - can be pretty intimidating. There is a real chance you can injure yourself. But you feel so great when you conquer your fears! It's the same when you risk a big change in your life. Quitting your job and moving abroad or pursuing a new career takes some guts. But if that really appeals to you, you just have to risk it - or risk remaining unsatisfied with your life!

第三尝试新事物。当你学习或者尝试新事物的时候总是会有一定的风险。滑雪伞甚至是学溜冰这些看起来都挺吓人的。确实你可能真的会伤到自己。但是当你克服了恐惧之后你会有很棒的感受闱理在人生中你冒险做出改变的话也是如此。辞职移民追求新的事业这些都需要勇气。但是如果这就是你想要的你就必须冒险然你这辈子都不会对自己的人生感到满意！

Number 4 - Risk Missing Out on Something. People don't like missing out on fun events or pleasant pastimes. Sitting in your favorite bar and chatting with friends every night or going on that vacation holiday can be very tempting. Devoting ourselves to our health, work and success makes it seems like we might be missing out. Business before pleasure can be uninspiring. But think about it for a moment: how many parties, concerts and beach holidays have really made a difference in your life? What could you achieve if you were willing to miss out on something, now and then?

第四承担错过一些东西的风险人们不希望错过好玩的事情或者令人快乐的消遣时光。每晚都坐在最爱的酒吧里跟朋友聊天或者去度假这些都很令人心动。花时间去管理我们的健康努力工作追求成功这可能会让我们错过一些东西。先苦后甜这是并不是什么鼓人心的事。但是想一想有多少派对需要会，在海边度过的假日能够让你的人生有所不同如果你愿意偶尔错过一些东西最后你会取得什么样的成就？

Number 5 - Risk Saying I Love You First. Being the first to say I love you to someone can frighten the bravest of us. It's because it makes you extremely vulnerable emotionally. And, if your partner doesn't say, I love you too, you will feel what seems like the ultimate rejection. Yet, that person could say those words you want to hear! If you feel that way, he or she very likely does as well - but they just may be more afraid of saying it than you are. And if they don't respond as you hope, at least you know where you stand and can reevaluate your relationship.

第五冒险当那个先表白的人。主动说我爱你这可能是大多数人都不敢做的事因为这让你在情感上显得很脆弱。而且如果对方没有回应说我也爱你你会觉得人家就是完全拒绝你了。然而对方有可能会说出你想要的回答如果你有那样的感受那么对方可能也会有一样的感受只是人家比你更害怕把那句话说出来而已。如果他们给的回答不如你愿起码你能知道目前的情况如何你也能重新审视你们的关系。

Number 6 - Expressing Your Opinion. Everyone has their own opinion on just about anything. Expressing your own point of view might seem risky. You don't know how it will be received. You might really upset someone. You might even find out that your opinion is demonstrably wrong! A lot of people prefer to avoid potential conflict or contradiction - and they don't speak up for themselves. Yet, if you don't say what you think - at least some of the time - others will think that you have nothing to offer or have no principles - or even lack intelligence. Interesting people express their opinions, even if they are sometimes unpopular. They are often successful people, because they have as little fear of failure as they have in saying what they think.

第六表达你的观点。对于每一件事每个人都有自己的观点。表达自己的观点，貌似这也是有风险因为你不知道自己的观点会得到什么样的回应可能你会让某个人不开心可能你甚至会发现自己的观点明显是错误的很多人希望避免潜在

的冲突所以他们不表达自己的观点。然而如果你从来都不表达自己的观点那么别人可能会认为你没有什么想法或者认为你没有原则甚至认为你不够聪明。那些有趣的人就算他们的观点可能不太受欢迎他们也会表达自己的观点。这些人往往是成功人士因为他们不害怕失败所以也不害怕说出自己的想法。

Number 7 - Risk Losing Friends. Friends are important. Life without them would be very dissatisfying. But they can also get in the way of your dreams. When you are pursuing your goals, you will often need to spend a lot of time by yourself. And friends can make it difficult for you to concentrate and remain focused. Some friends will even try to get you to give it all up because they are feeling left out. True friends will take your dreams seriously and understand that you need to take some time away from them. They will work on preserving their friendship with you, even if they don't see you as often as they would like.

第七承担失去朋友的风险朋友是很重要的没有朋友的人生是难以令人满意的。但是朋友也可能成为你逐梦路上的绊脚石。当你追逐梦想时你需要独自度过很多时间朋友会让你难以专注。有些朋友甚至会尝试让你放弃梦想因为他们觉得自己被抛弃了。真正的朋友会认真对待你的梦想他们理解你能够接受你和他们待在一起的时间变少。就算和你见面次数比他们期待的少他们也会努力维持这段友谊。

Number 8 - Risk Inadequacy. Sometimes you will not be good enough. Sometimes you won't have what it takes. But isn't it better to find that out for yourself than having to ask yourself if, maybe, you could have become a great musician or a sports star? Sometimes you have the talent or skills, and sometimes you don't. You might become obsessed with a certain 'what if' your entire life. But if you knew you could have never made it, you won't be haunted by useless regrets. You will never know unless you give it your best try and risk not being good enough.

第八冒险做一个有缺陷的人。有时候你就是不够优秀你就是没有足够的能力。如果我能成为一个伟大的音乐家或者足球明星会怎样你这样质问自己还不如自己去发现你还不够优秀或能力不够的事实对吧或许你这一辈子都会纠结某些可能做到的事但是如果你明确知道自己永远做不到你就不会一直后悔，这种后悔是没有意义的。只有当你真的全力以赴了并且冒险做一个不够优秀的人你才会看清真相才会明白这个道理。

Success is a quiet process

As Ellen Cook puts it, success is like a quiet daily set of tasks. Real real small. It's like that quiet walk to the library, that empty 24/7 library late at night, over and over and over. Or as I sit there studying other great people and I compare their actions with my own actions over and over and over. Or as I sit doing 30 minutes of meditation a day over and over and over. Making the choice to eat foods that enhance my brain neuro-transmitters over and over and over.

正如Ellen Cook所言成功就像每日默默成的任务。这些任务很小比如每天晚上都自己走去那个空荡荡的24小时图书馆或者我反复地研究其他优秀的人并把他们的行为和我的进行对比或者我每天都用 30分钟练习冥想或者我一直坚持吃健康的、有益于我的大脑神经递质的食物。

It's a very quiet process where you're doing these simple little tasks but finding love in those simple little tasks. It's not the big thing where you do this one thing and something big happens. Being able to make a success out of your life. It's not purely down to luck. You have more control over it than you think. You can decide to be successful. You can decide to have the fast cars and the big house.

这个过程很安静的任务很简单也很小但是你逐渐喜欢上它们。它们不是什么伟大的事情不是那种影响深远的事情。在人生中获得成功并不全靠运气。你对生活的掌控力比你想象的要强。你可以决定自己是否成功是否能够拥有豪车和大房子。

And it starts right here with your studying, not because getting good grades is going to bring you success in the future, it's about the work ethic and the attitude you have towards your studying that will transfer to your work after you graduate. It's about believing that you can achieve something and doing whatever it takes to achieve it. It's about fine-tuning the skills you have to zero in on your goals, the skills that you are improving while you're studying.

这一切都是从现在开始的从你目前的学习开始这不是因为好成绩能让你在未来取得成功而是因为你对学习这件事的职业精神和态度。在你毕业之后它们仍然能够影响你。这也是因为你相信自己能够实现梦想并且用尽全力为之奋斗。这更是因为你不断打磨自己的技能从头学起通过学习来不断提高它们。

Work ethic, time management, reading, critical thinking, problem solving, decision making, focus, reasoning, persuasion, organization, overcoming obstacles, and self-motivation. These are all skills that all great students have improved and refined over years of efficient and effective studying. And it's these students that have made a decision that when the world is saying that's impossible, you're not capable, your dreams will stay as dreams.

职业精神时间管理阅读批判性思维解决问题能力做决定的能力专注力推理能力口才组织能力克服困难能力自己给的动力这些都是优秀的学生们在多年的高效学习中提高能力。在全世界都跟他们说不可能、说他们做不到、说他们无法实现梦想的时候他们能够做出自己的决定。

They have that lone voice that goes, no, you know what, it is possible, just watch me. They have gone through this process of growth, and they have pushed and pushed through all the obstacles that were thrown at them. It's important that you don't think of your studying as though it's all about passing your exams and getting a piece of paper at the end of it all so you can get a good job, because it's so much more than that.

在他们心中有一个声音响起。不十切皆有可能走着瞧吧他们经历了成长的过程无论遇到什么样的困难们都能一一克服。你不要认为学习就是为了通过考试、拿到文凭、得到好工作。这一点很重要因为学习的意义远不止于此。

And it's the great students that understand this. It's not about your exams. It's about the process and the journey you go on and the skills you refine over the years. It's about when you don't feel like studying but you continue to push through and study anyway. It's doing the things that you need to do, that you know you need to do when you don't feel like doing it, because the reality is, is that most people can study when they're feeling good.

优秀的学生才会明白这个道理。学习不是为了考试学习的意义在于它的过程本身在于你这些年来练就的技能。学习就是在你不想学的时候还能督促自己去学习。学习就是要去做那些你需做却不想做的事情。因为实际对于大多数人来说在自己状态好的时候学习不是什么事。

Most people can study when they have no distractions, when they're in a quiet environment, when there's pressure put on them as a deadline is approaching. But just being able to study on the highs isn't good enough. You have to be able to study all the time. So learning how to refocus your mind on the lows and knowing the lows are going to end is a huge skill. The process is just as important as the actual prize, because the process is going to make you. The deeper the process, the greater the reward. Just remember the prize is going to be huge.

在没有令自己分心的事情时在环境很安静在没有截止日期逼近的压力时大多数人都能学习。但是在高峰时期学习这还不够。你要做到在任何时候都能学习。所以学会在低谷时期专注于学习并且告诉自己艰难日子终将过去这是很重要的能力。过程和结果一样重要因为过程最能塑造你。过程越深刻收获越丰富。记住你将会获得巨大的收获。

Whats On Your Life List?

What is a successful life?

什么是成功的人生？

To answer this question, we must observe the paradox of our priorities. How we pursue lifestyles built on our resumes. Yet as we knowingly approach death, we yearn to be remembered by every quality, except that resume.

要回答这个问题，我们必须研究一下我们做事的优先次序。我们在简历的基础上，追求自己的生活方式。 但是当我们生命垂危之际，我们却渴望被人们铭记自己简历之外的所有品质。

You see, during the course of our lives, we seek success of every tangible sort. But when it comes to looking back on a life once lived, those tangible things, the measuring cups of our entire existence seem emptier than ever. I have never read nor have I written a eulogy that expressed how accomplished a person was or how brilliant of a worker they were. Even the most celebrated innovators are not eulogized by the impact of their inventions, but by their motives, the bonds they formed, acts of kindness, and what they meant to the people around them.

你看，我们活着的时候追求的是各种有形的成功，但是当回首往事时，那些有形的东西以及证明我们生活过的事物，此刻却像量杯一样空空如也，比以往任何时候都更空虚。我从未读过或写过因一个人成就巨大或技艺娴熟而被人颂扬的悼词。即便是最著名的发明家，人们津津乐道的并非他们的发明，而是他们的动机，因他们而形成的纽带，他们善意的行为，以及他们对周围人的意义。

So if you wonder what a successful life really feels like, all you have to do is close your eyes and think about what you want your eulogy to sound like. What words will be used by loved ones describing and celebrating your legacy when you die? List those words. Internalize this list, memorize it, stare at it everyday . What you're staring at is your personal definition of success.

所以，如果你想知道成功的人生是什么样子的，你只需闭上眼睛，想象一下你的悼词内容。在你死后，你的亲人们会如何形容和赞美你的遗赠？把那些词语列成清单，放在心上，记在脑子里，而且要每天都盯着它。因为你所注视的恰恰就是你个人对于成功的定义。

It sounds crazy , but everything else you're pursuing, everything else you whine and worry about is secondary. And if you're going to do those secondary things, make sure they serve your list. Make sure they increase your capacity to live by those qualities. Stop chasing things that aren't on your list. Because that list is all you have and that, my friends, is the only way to live a successful life with intent, not regret.

这听起来很疯狂，但其他你所追求、抱怨、担忧的任何事情都是次要的。并且，如果你要做这些次要的事情，在做之前要确保这些事对完成你的清单有帮助，要保证它们可以提高你的能力，从而让你具备那些品质。要果断放弃那些不在你清单上的事情，因为这张清单就是你的所有，而且这是你成功的唯一途径，否则你将抱憾终生。

What's on your list?

你的清单上有什么呢？

You Don't Know How Strong You Are

Tough times separate those who give up and those who keep going. Tough times show what heart is really inside. Tough times show what courage is really inside. It's in our toughest moments that we see who people really are. Who are you? "You don't know how strong you are, until being strong is your only choice." Bob Marley said this. And it is so true! You don't even know how strong you are! You have inside you something so great, something so deep, something so strong. Do find that strength inside you.

艰难岁月把轻易言弃的人和奋勇向前的人区分开来。困苦的时刻揭露人的内心展现了人的勇气。正是在最艰苦的时候我们才能看清世人。你是谁在坚强是你唯一的选择时你才知道自己有多坚强。这是 Bod Marley说的说得对极了！你甚至根本不知道自己到底有多强大在你的内心深处你拥有很伟大、很深刻、很坚强的品质。你一定要找到这内在的力量。

Whatever you are going through, you can get through it. You will get through it, and it will make you stronger. It will make you wiser and it will make you better. That is what mentally tough people do. They do not allow anything to break them. They force hard times to make them. They learn lessons. They see the blessings. And most importantly they keep going. What lessons can you learn? What blessings can you take from this? If you asked me what is the most important thing one can do in overcoming anything. In one word I will give it to you. Purpose.

无论你正在经历什么你都能渡过难关。你会战胜困难且这会使你更加坚强。它会使你更睿智、更优秀。内心强大的人会怎么做他们不会让任何事打败自己，他们用困难塑造自己他们吸取经验他们看见了困难那宝贵的财富。最重要的是他们永不放弃。你学到了什么经验从中得到了什么样的收获如果你问我对于一个人而言要克服困难重要是什么我只会告诉你一个词目标。

If you know your purpose, if you know why you do what you do, if you know why you must fight, then you will fight. You will fight for your purpose. You will fight for your love. You will fight for that one person. You will fight for your family, for your friends. You will fight for your legacy. You will fight because you know you don't want future generations to give up, because you did. You want them to keep going because you did. You want them to say I will fight because I saw you fight. I will stay strong because you stood strong. That is mental toughness!

如果你知道自己的目标如果你知道你为什么做你所做的事如果你知道自己为什么必须斗那么你就会去奋斗。你会为了你的目标而奋斗为了你的朋友而奋斗为了你的财富而奋斗。你会奋斗因为你明白你不想因为自己的放弃而使得后人做事也半途而废。你想让他们不断进取因为你做到了。你想让他们说我会奋斗因为我看见了你的奋斗我会保持坚强因为我看见了你的坚强。这就是精神的强大！

Life Is Like An Arena

Life is not a ride in a wonder park, where thrills and challenges are there only to excite you without a real intention of defeating you. Life is more like an arena where you must fight to survive. You are either defeated or a victor.

在游乐园里那些惊险和挑战不是为了打倒你而是为了让你感到兴奋但在生活中可不是这样。生活更像是一个竞技场你要拼搏厮杀才能生存。你要么赢得光荣要么输得头破血流。

What defines defeat is not being knocked down on the ground, but not having the will to stand back up, because if you're breathing, life still believes in you, and you still have a chance of standing back up again, and showing the people and circumstances that you are above and bigger than them. Everyone has their unique problems in their life, their own mountains to climb, a climb that never gets easier. It's you who gets used to them. Get used to the challenges and problems in your life and pick them out one by one.

什么是被打败被打败不是被打倒而是没有站起来的勇气因为如果你还活着生活还是有希望的你还有机会再次站起来让世人知道让你的苦难知道，你比他们更强大。每个人都有他们自己独特的问题们有自己的攀要攀爬这绝壁事。你要适应这些。适应挑战适应问题个个解决他们。

Fearing your problems is not a solution. It's an impulsive response that forbids you from challenging it due to the fear of defeat. What good is that saved sense of being undefeated when you haven't even tried to beat the biggest of your life's monsters? At times when you are struck by life's biggest challenges, you might ask yourself, why me? Why did life choose to hit me with the hardest punch? The answer that you should be giving yourself is that it's because you are the only one who can bear it and still survive. Not survive but thrive in this world.

恐惧问题能解决问题恐惧只是一种下意识的反应而已它会让你因为害怕失败而不去挑战问题如果你连尝试打败困难的勇气都没有那么那种不被打败的感觉又有什么意义呢有时候你被眼前的巨大挑战困住了你可能会问自己为什么是我为什么生活要让我遭受最重的打击你该告诉自己这是因为只有你能够承受这些并从中生存下来。不仅是生存下来而且是活得精彩。

No known warrior became known to leave his name in the pages of history without having scars on his body. The problems in your life are injuries that when overcome would merely become scars of pain and suffering on your body, scars that would remind you of the lessons you have learned, mountains who have conquered, and enemies that you have defeated. So, keep that in mind and keep charging against the enemies, because no matter how strong and tough your problems boast to be, beat down they are always afraid of you.

那些名垂青史的勇士他们的身上都有疤痕。你生活中的问题是你会受的伤，它们留下的疤痕会一直提醒你你曾经吃过教训你曾经爬上巅峰你曾经打败死敌。所以记住这一点不断回击你的敌人因为无论敌人有多么强大无论困难有多么棘手它们其实都很害怕你。

It's your fear that encourages them to even challenge you. Having the right mindset. It doesn't sit back and hope for the problems to solve themselves, or the storm to go away on its own. There is never a problem in reality. It's your approach towards it that makes it a problem or an opportunity. And lf you have time to whine and complain about something, then you have the time to do something about it,"Anthony JD Angela.

你的恐惧让它们有勇气去挑战你。保持正确的心态这不意味着坐视不管等着问题已解决也不意味着等着乌云自动消散。没有真正的问题对待事情的方式决定了你面的是问题是机遇。正如 Anthony JD Angela说的那样如果你有时间发牢骚时间抱怨那你也有时间去行动。

All problems have one thing in common. They all demand a solution for themselves. And if you don't give them the solution, they'll keep bothering you until your dying day. You should remember that you are not the only person who is facing problems. Neither are you the only one who manages to solve them. and solving a problem is much easier than living through the never-ending pain and agony of having an unsolved problem.

所有的问题有一个共同点它们都需一个解决办法。如果你不给出解决办法它们就会永远地不断地烦扰你。你应该记得面问题人不止你一个解决问题人也不止你一个。比起那种还未解决问题带来的永无止尽的痛苦解决问题得简单多了。

All it requires from you is commitment towards solving the problem, dedication and hard work. You can be anyone and anywhere you want to be in your life. All it requires from you is hard work. So, are you willing to give what it takes to be what you desire? If yes, then success is waiting to be your slave forever.

你需做的就是下定决心解决问题心致志付出努力。你可以变成任何人去到任何的地方。你只需投入很多努力就好。那么你能否付出足够的努力去成为自己想成为的人如果你的答案是肯定的那么成功将永远伴你左右。

Life is too short to live someone else's dream

Larry Ellison甲骨文软件公司 CEO身价 700亿美元。他在32岁以前还一事无成开始创业时只有 1200美元却使得 Oracle公司连续12年销售额翻一番成为世界上第二大软件公司他自己也成为硅谷富翁。

Growing up in a low-income family on the south side of Chicago, virtually everyone important in my life, my family, my teachers, my girlfriend, wanted me to be a doctor. Over time, their dream became my dream.

芝加哥南部的一个底层中产阶级社区这就是我长大的地方。在我的生命中几乎所有重要的人我的家人我的老师我的女友都想让我成为一位医生。久而久之他们的梦想变成了我的梦想。

They only needed me I should be a doctor, but as hard as I tried, I could not do it. I was unable to make myself into the person that I thought I should be, so I decided to stop trying. I was 21 years old when I dropped out of college.

他们让我相信我应该成为一位医生。但是无论我多么努力我都做不到我无法变成理想中的自己。所以我不再尝试这样做。那时我 21岁从大学辍学了。

During my California springs and summers, I spent most of my days in the High Sierras in Yosemite Valley, working as a ranger and a rock-climbing instructor. I loved those jobs, but unfortunately, they didn't pay that well. So, I also got a job working a couple of days a week as a computer programmer back in Berkeley.

在加州度过的那几年太多数时候我都待在约塞米蒂峡谷的西亚拉山区我在那儿当河道向导当攀岩教练。我爱这些工作然而不幸的是它们带来的薪水并不高所以我也在当伯克利当程序员每周在那儿工作几天。

I had learned programming in college. I didn't love programming but it was fun and I was good at it. I started taking classes at UC Berkeley. I took several classes, but the only one I can remember was a `sailing class` taught at Berkeley `marina`. When my class was over, I wanted to buy a `sailboat`.

我在大学的时候学过编程我不喜欢编程但编程还挺有趣的而且我很擅长编程。我开始在加州大学伯克利分校上课我选了几门课但是只有一门课让我记忆犹新那就是在伯克利码头那里上的帆船课。这门课结束后我想买一艘帆船。

My wife said this was a single stupidest idea she had ever heard in her entire life. She asked me of being irresponsible, and she told me I lacked ambition. She kicked me out, and then she divorced me. This is a `pivotal` moment in my life.

我的妻子说这是她听过的最愚蠢的想法。她指责我说我不负责任说我胸无大志。她把我赶出家门然后和我离婚了。这可是我人生中的关键时刻。

My family was still angry at me for not going to medical school, and now my wife was going to leave me because I lacked ambition. It looked like a repeat of the same old problem. Once again, I was unable to live up to the expectations of others, but this time, I was not disappointed in myself for failing to be the person they thought I should be.

我的家人还在生我的气因为我不学医了。我的妻子要跟我离婚因为我没有抱负。这一切就像在重蹈覆辙。我无法满足别人的期待这已经不是第一次发生了。换做以往我会对自己感到失望因为我没能成为他们认为我该成为的人然而这一次我不再对自己失望。

Their dreams and mine were different. I would never confuse the two of them again. Though in my 20s, I continued painting, trying different things, racing bikes and boats and constantly changing jobs. I searched and searched, but just could not find a software engineering job that I thought that I loved as much as I loved sailing. So, I tried to create one.

他们的梦想和我的梦想是不一样的我再也不会混淆这两者。在二十多岁这段时光里我不断试验尝试不同的事情骑自行车划船不断换工作。我很热爱帆船运动然而这些年里我找了很久都没有一份编程工作能让我感受到同样的热爱。所以我尝试自己创造一份这样的工作。

My goal was to create the perfect job for me, a job that I loved and never expected the company to grow beyond 50 people, so, maybe I really didn't lack ambition or vision back then. The only goal I always had was 50 people, but when I started, it was not my intention to build a big company.

我的目标就是为自己打造一份完美的工作一份我真心热爱的工作。我从来没想过我的公司规模会超过 50人。所以也许当时我确实是没多大野心也没什么远见。如今我们公司有十五万员工。但是在创业初期我没有想过要把公司做得这么大。

We assembled an all-star team of gifted programmers, who were among the best in the world at what they did. That team's one crazy idea gave birth to a giant company. I call it a crazy idea, because of the time everyone told me it was a crazy idea. The idea was to build the world's first relational database.

我们把很有才的程序员们聚到一起组成了一个全明星团队他们是各自领域里的顶尖行家。一个优秀的团队加上一个疯狂的想法就产生了一个超大规模的公司。我把它称之为疯狂的想法是因为曾经所有人都告诉我想创造世界上第一个关系数据库这个想法很疯狂。

But back then, the `collective` wisdom of experts was, that while relational databases could be built, they would never be fast enough to be useful. I thought, all those so-called experts were wrong, and when you start telling people that all the experts are wrong, at first, they call you `arrogant`, and then they say you're crazy.

但是当时电脑专家们都认为要建一个关系数据库可以但这个数据库的运行速度会很慢所以它没什么用处。我认为那些所谓的电脑专家们他们的看法都是错的。当你告诉别人说专家们都是错的人们就会说你很傲慢然后他们会说你太疯狂了。

So, remember this, graduates, when people start telling you that you're crazy, you just might be onto the most important innovation in your life. Our doubled in size year after year after year for 10 years. It was growing so fast that it was impossible for anyone to control. It was like sailing in a hurricane. And then we went public.

所以毕业生们请记住如果有人跟你说你很疯狂那么你可能正要经历人生最重要的变革我们公司的规模连续十年实现翻倍。它发展得非常快势不可挡它就像是在暴风中航行一样。然后我们就上市了。

Oh my god. Maybe I should have been a doctor.

噢也许我本应该当个医生！

NW ear, NW ou

ts that tie of year again, the new year , tie for some new year esoltions. But not so fast. Stes have show 8% of pole who set new year esoltions fail to eab them

又是新的一年每到这个时候人们都会下定决心要在新的一年做成一些事情。且慢研究表明， 88%的人虽然下定了新年决心但却没能实现。

h fat, the apity of those pole will have it before we are in the seond mth of the year. Less than 2 so thing ifeent. They stik to their goals. That's the statistis. 8 fail and 12 stik at it. And that desnt even an the esults of the 12 are lasting.

事实上太部分人还没到第二个月就已经放弃了。只有 12%的人没有放弃而是坚持朝着目标努力。这就是数据呈现的情况，88%的人失败了放弃了， 12%的人坚持下去了。而且这并不意味着这 12%的人取得的成果能一直保持下去。

Why d that any pole fail? And more iptantly , why d the so few sees? Thee are 5 min easons. Listen in so you an be one of the few who one ur their goals, not just new year goals, but ey kind of goals, tagets and dams you have.

为什么这么多人都失败了更重要的是为什么成功的人这么少原因有五个请认真听一听这样你就能成为实现目标的少数人不仅是新年目标还有各种各样的目标和梦想。

***Number 1*: Ee you year , take in the good, notie the blessings and se the lessons. This is a pwful pess, not just in looking fowad to lean and abiev more, but also looking bak at how far you ve come and being gateful for ey thing that was great this past year .**

第一回顾过去的一年吸收好的东西多关注那些美好的事并学会应用你学到的教训。这个过程会产生巨大的影响不仅能帮助你朝前看、取得更多成就而且能帮助你回顾自己走了多远、对过去一年所有美好的事心怀感激。

Get a pen and pap and we dw you answs to the following qstions. We will pid a dwload sheet in the desiption as well. Think bak over the past year .

拿一支笔写下你对以下问题的答案我们也会在视频介绍里提供一个可以下载的文档。请你回顾思考过去的一年。

Wite dw: What was great this past year? What was you favite emy? What are you proud of? What was you greatest abievement? What was you greatest lesson? What was unaeptable for that year , in other ds, what an you d better? What lessons old you take into this nex year to ake sure you are in a better position in 12 mths' time?

请写下在过去的一年中很棒的事情有哪些你最喜欢的回忆是什么你为什么感到自豪你最大的成就是什么你最大的教训是什么你觉得难以接受的事情是什么或者说你能把什么事做得更好你能把什么经验带到下一年、确保自己在

新的一年结束时有更好的结果？

Number 2:Do you know where you're going? Most people give up on their goals and resolutions so easily because they don't have a clear target they are working toward. They have vague resolutions like, lose weight, or , make more money , but no plan as to how what will be achieved and no deadline for when it will be achieved by .

第二你知道自己要去往哪里吗很多人轻易地放弃了他们的目标和决心因为他们的目标不清晰。他们的决心很模糊比如减肥挣更多钱但是他们没有计划好如何达成这些目标截止期限是什么。

If it is not clear , it will not happen. If it is not planned, it will not happen. Get clear about the result you want have. For example, I will lose 30 pounds, I will make one million dollars in extra income

如果目标不够清晰那它就不会实现。如果没有做好计划那它就不会成真。想清楚你到底想要什么样的结果。比如我要减肥 30磅我要额外多挣一百万美元。

Get precise about the time in which you will achieve it. I will lose 30 pounds, by March 30th, 2020 I will make one million dollars in extra income by December 31st, 2020. Get clear about what you want and when you want have it. And you will have it.

为你的目标设定精确的截止时间。我要在2020年的3月30号之前减去30磅体重我要在2020年12月31日之前挣到一百万美元的额外收入。你要想清楚你想要什么什么时候要做到。这样你就会得到你想要的。

Number 3:Change your identity . For many people their goals and resolutions fail because, although they might have a plan, they never believe they are that person that can change long term

第三改变你对自己的看法。对于很多人来说他们的目标和决心无法达成是因为虽然他们也许有计划但是他们从来不相信自己能做出改变、并长期坚持。

The person doesn't lose weight because they identify themselves as a sweet tooth. The person doesn't quit smoking, because they identify themselves as a smoker and they still believe they need it for stress relief. You must change how you see yourself. If you want permanent change, it's to your old self so your new self can be reborn.

人们无法减肥因为他们认为自己就是嗜甜如命的人。人们无法戒烟因为他们认为自己就是个烟民并且他们相信他们需吸烟来缓解压力。你必须改变你对自己的看法如果你想做出长久的改变那就要跟过去的自己说再见这样才会产生新的自我。

Number 4:Love yourself. If the people around you are not uplifting the positive changes you want to make in your life, distance yourself from these people, until you make the changes you want make. If the same people are still not supportive after you've made these great changes, maybe you need to review the people you are around

第四改善自己的圈子。如果你身边的人不支持你做出积极改变那么请你远离这些人直到你成功地做出了改变为止。如果到了这时这些人仍然不支持你也许你需要反思一下你身边的人。

Surround yourself with people who are going to help and support you, people who want to see you make these changes permanent. If they are not in your immediate environment, find them online. Follow positive people, people who have already achieved the result you want to achieve. Listen to their podcasts, read their books. Immerse yourself in the result you want to achieve.

多接触那些会帮助你、支持你的人那些想要看到你做出改变、并长期坚持的人。如果你身边没有这样的人去网上找找看。关注那些积极向上的人那些已经达成了你的目标的人。听他们的播客读他们的书让自己沉浸在你想获得的结果里。

***Number 5*: Believe. Believe it is possible by finding stories of others who have done it before you. Believe in yourself by taking the first step, then another and another and being aware and proud of your progress. Believe you deserve this, because you do.**

第五要心怀信念。请你多找找前人的成功实例让自己相信一切皆有可能。你要迈出第一步第二步第三步关注你努力的过程并为此感到自豪这样你就会对自己有信心。你要相信你值得拥有这些因为你确实值得拥有它们。

Don't take your mistakes from last year into this year. Use the pain. Learn from it. Grow from it. Set a new standard for your life and commit to make this coming year your best year yet! It's time to step into the new you.

不要把你犯过的错误带到新的一年。利用你经历过的痛苦从中学习收获成长。为自己的生活设立新的标准努力让新的一年成为你人生中最好的一年是时候去拥抱新的自己了！

The biggest mistake is you think you have time

When asked "What's the biggest mistake we make in life?" the Buddha replied "The biggest mistake is you think you have time."

当被问及"人生最大的错误是什么"佛说："最大的错误是你认为你自己有时间。"

Time is free but it's priceless.

时间是免费的但也是无价的。

You can't own it, but you can use it.

你不能拥有它但你能够使用它。

You can't keep it but you can spend it.

你不能储存它但你能花费它。

Once it's lost, you can never get it back.

一旦它消失了你就再也拿不回来了。

That average person lives 78 years.

人的一生平均有78年。

We spend 28 years of our life sleeping.

我们花28.3年在睡觉上。

That's almost a third of our life but 30% of us struggle to sleep well.

差不多花了我们三分之一的时间但同时 30%的我们在为睡个好觉而斗争。

We spend 10 years of our life working but over 50% of us want to leave our current jobs.

我们花10.5年去工作但超过 50%的人想离开现在的岗位。

Time is more valuable than money .

时间比金钱更有价值

You can get more money ,but you can never get more time.

你会赚更多的钱却不能赚到更多的时间。

We spend 9 years on TV and social media.

我们在电视和社交媒体上花费了9年的时间。

We spend 6 years doing chores.

花6年的时间在家务上。

We spend 4 years eating and drinking. 四年的时间花在吃吃喝喝上。

We spend 3.5 years in education.

累计消费3.5年在教育上。

We spend 2.5 years grooming.

2.5年花费在打扮上。

We spend 2.5 years shopping.

2.5年花费在购物上。

We spend 1.5 years in childcare

1.5年的时间花费在呵护孩子。

And we spend 1.3 years otting.

1.3年的时间花费在通勤上。

That leaves us with 9 years.

剩下留给我们的只有9年的时间。

How will we spend that time?

我们将会怎么消费这些时间呢？

Steve Jobs said your time is limited so don't waste it living someone else's life."

乔布斯说生命有限，将它浪费在重复他人的生活上。

So there's good news and there's bad news.

所以这有好消息也有坏消息。

That bad news is time flies; the good news is you're the pilot.

坏消息是时光飞逝，好消息是你是航行者。

Imagine you wake up every day with $86,400 in your bank account.

想象一下你每天醒来银行账户中都有86400美元。

And at the end of the night, it's all gone whether you spent it or not.

一天结束它们都消失了不管你是否消费它们。

And then the next day you get another $86,400.

第二天你的账户又会拥有86400美元。

What would you do with it?

我们将会用它做什么呢？

Every day 86,400 seconds are deposited into your life account.

每天86400秒都会存入你的生命账户。

And at the end of the day one they're all used, you get a new 86,400 seconds.

一天结束后第二天你将拥有新的 86400秒。

We would never waste it if it was money ,so why do we waste it when it comes to time?

如果是钱我们绝不会浪费那为什么我们要浪费时间呢

Those seconds are so much more powerful than dollars because you can always make one dollars,you can't always make one time.

时间比金钱的力量更大因为你能赚钱却不能赚取时间。

To realize the value of 1 year ,ask a student who failed a grade.

想知道一年的价值要去问考试失利的学生。

To realize the value of 1 month,ask a mother who lost their child in the final month.

想知道一个月的价值去问在分娩前一月失去孩子的母亲。

To realize the value of 1 week,ask the editor of an online magazine.

想知道一周的价值去问在线杂志的编辑们。

To realize the value of 1 hour ,ask the people who' s in a long distance relationship

想知道一小时的价值去问异地恋的情侣们。

To realize the value of 1 minute,ask the person who just missed a bus,train or plane.

想知道一分钟的价值去问刚刚错过公交和飞机的人。

To realize the value of 1 second,ask the person who just missed an accident.

想知道一秒钟的价值去问刚刚幸免于难的人。

To realize the value of 1 millisecond,ask the person who just came 2nd at the Olympics.

想知道一毫秒的价值去问与奥运冠军失之交臂的运动员。

We think it's people wasting our time but it' s really us giving them the permission to do that.

别人总是浪费我们的时间其实是我们准许他们这样。

And in reality ,these two people live inside us.

现实是我们体内有两个人。

Don't let someone be a priority when all you are to them is an option.

当你对别人只是可选项时,别将他们排在前面

Some of us lose the people most important to us because we don't value their time.

我们失去了很多重要的人因为我们没有珍惜他们。

Some of us don't recognize how important someone is to us until they're gone.

有时直到失去才知道他们有多重要。

Inside all of us are two voices.

我们的内心有两种声音.

One voice that wants to uplift, one voice that wants us to expand, one voice that wants us to grow.

一种声音鼓励我们上进,鼓励我们丰富自己,鼓励我们不断成长.

And then there's the other voice. The voice that holds us back, the voice that makes us lazy, the voice that makes us complacent. The voice that restricts us from our potential.

还存在另一种声音----这种声音使我们退步让我们变得懒惰让我们骄傲自满。这种声音限制了我们的潜能。

Every day from the moment we wake up until the moment we go to sleep inside of us there's this battle between the two voices.

每天从醒来到入睡这两种声音都在斗争。

And guess who one wins?

猜猜谁赢了？

The one that we listen to the most.

我们听得最多的那个声音赢了。

The one that we feed.

我们更喜欢的那个声音.

The one that we amplify.

那个让我们更强的声音.

It is our choice of how we use our time.

我们选择如何使用我们的时间。

Life and time are the best two teachers.

生活和时间是两位最好的老师。

Life teaches us to make good use of time.

生活教会我们更好地使用时间。

And time teaches us the value of the life.

时间教会我们生活的价值。

And as William Shakespeare said,

就像莎士比亚说过的一样，

"Time is very slow for those who wait, very fast for those who are scared, very long for those who are sad, and very short for those who celebrate , but for those who love, time is eternal."

"对想要它的人来说时间很慢对害怕它的人来说时间很快对悲观的人来说时间很长对充满欢愉的人来说它又很短暂 ,但对于爱它的人来说时间即是永恒。"

The Easy Road or The Hard Road

Your educational journey is going to be hard. I can promise you that. There are two paths you can take, the easy road or the hard road.

我可以很确定地告诉你你的求学过程将会困难重重。你有两条路可选容易的路或者困难的路。

If you choose the hard road, it means you choose to study when your friends are out partying. You start writing your report when the rest of the class don't even know the deadline yet. You're already awake at 5: 00 a.m. on your way to the library when the rest of your friends are still sleeping.

如果你选择了困难的路这就意味着朋友们尽情欢乐时你要选择努力学习其他同学还不知道截止日期时你已经开始写作业凌晨五点朋友们还在沉睡时，你已经起床出门朝着图书馆去。

If you can find enough focus and motivation to do these hard things, the things that everyone else struggles with, then your educational journey will be easy.

如果你有足够的专注力、足够的动力来做这些让大家觉得痛苦的事那么你的求学过程将会变得容易。

The grades will be easy. The scholarships will be easy. But it's a choice that you have to consciously make. You have to step up and take that path. And it will be lonely at times. It will feel like you're the only one doing this.

成绩很容易变好奖学金很容易到手。但是你要自觉地做出这个选择。你必须加快脚步踏上这条道路。有时你会感到孤独好像这些事只有你自己在做。

But that's the whole point. The whole idea of this is to be standing at the top of the mountain, while everyone else is still at the bottom, because you took the path that so many people are unwilling to take.

但是这才是有意义的地方。你这么努力就是为了站在顶端其他人只能仰望你因为你选择了别人不愿选择的路。

Okay, so you know where you want to be, at the top of that mountain, but why aren't you there yet? Why is it that you're stopping yourself from getting there?

所以你已经很清楚自己想成为站在顶端的人但是为什么你还没做到？

Why is it that you just settle for the grades that you're getting? Because doing well at school and university, it's not easy. It's difficult to get good grades. It takes a lot of focus, a lot of sacrifice, a lot of late nights and early mornings.

为什么你一直无法达到这个高度为什么你满足于现有的成绩因为在求学生涯中无论是小学、中学还是大学要想变得优秀都是有难度的。想取得好成绩，这不容易。你要很专注要牺牲很多要早起晚睡。

But it's absolutely worth it. You are in charge of the life you live, and when you're in charge, you study hard, regardless of how you're feeling, whether you're feeling lazy, tired, stressed, frustrated, bored, it doesn't matter. This doesn't stop you.

但是这一切都是值得的。你的人生你自己说了算。如果你能把握自己的人生，无论自己感受如何、心情如何都没关系。就算你感到懒惰、疲惫、压力大、沮丧、倦怠也都没关系。这一切都无法阻你。

Use the P ins Motivation

You will come into obstacles. You probably already have, whether it's procrastination, friendship problems, family problems, money problems. Whatever it is, it's not going to be easy.

你会遇到困难。可能你已经遇到了比如拖延症友情问题家庭问题经济问题无论这困难是什么它都很难被破。

Pain is just a by-product. If you haven't experienced pain, then you will not have the motivation to become the best version of yourself. The pain is what will fuel you.

痛苦只是一个副作用而已。如果你没有经历过痛苦那么你就没有动力去成为最好的自己。痛苦会让你充满斗志。

The most successful people on the planet have had to overcome catastrophic obstacles, and they came out on the other side stronger and wiser.

在这个世界上最成功的人都经历了巨大的痛苦他们成功度过了难关变得更加强大更加睿智。

Bill Gates's first business failed miserably. Albert Einstein didn't speak until he was four years old. Both Walt Disney and Beethoven were given by their at 4 years old to a baby boy who died shortly after wards. Thomas Edison failed 1,000 times before creating the light bulb. Benjamin Roosevelt became partially paralyzed at the age of 39 for the rest of his life. He went on to lead the United States as one of the most respected presidents in history.

比尔盖茨他最初做的生意以惨痛失败告终。爱因斯坦他直到四岁才学会说话。欧普拉温弗莉她在十四岁的时候生了一个男婴但孩子没多久就夭折了。爱迪生他失败了一千次才发明出灯泡。富兰克林罗斯福他在39岁的时候遭遇了部分瘫痪余生都如此但他仍然坚持领导美国他是历史上最受尊敬的总统之一。

So it's not only you that is facing obstacles. It happens to everyone. But how you deal with them is what will set you apart from everyone else. And as they say after the rain comes the rainbow. There are two sides of pain that a lot of people don't really understand. There's a side of pain that most people can relate to, the difficult side, the uncomfortable side. You always remember what that feels like.

所以面对困难的不只有你太家都一样。但是你处理困难的方式会让你与众不同。正如人们说的雨后才有彩虹。很多人不明白痛苦也是有两面的。大多数人知道痛苦的其中一面就是困难的一面令人不适的一面永远都会记得这种感觉。

But then there's the other side of pain. It's called effort. It's called learning. It's called if you can push through the suffering, there's some great things waiting for you on the other side. But you don't get to see them if you give

right now ,andthat's why a lot of people don't get to see this side of pain, because they give up before they get there.

但是痛苦还有另一面是努力是胜利。它意味着如果你能熬过艰难你会收获很大的惊喜。但是如果你现在就放弃了你就无法看到这一切所以很多人没见过痛苦的这一面因为他们半途而废了。

It's not easy to continue studying when every part of your body is telling you to give up. It's so easy to give up. But it's at this point you need to give it everything you've got. It's about pushing yourself to the point where you feel out of your comfort zone. Because it's outside your comfort zone where the growth happens.

当你的身体想让你放弃时还坚持学习这是很困难的。放弃是很容易的但是到了这个时候你应该放手一搏。你要敦促自己直到你走出舒适区为止。因为只有走出舒适区你才能成长。

No one has been successful while staying within their comfort zone, absolutely no one. The first three seconds that you start studying, that's the hardest part. But if you can push through that, if you can push through the first few seconds of studying, it gets a lot easier .You realize it was the thought of studying that you were more bothered about than the actual studying itself.

获得成功的人都是走出舒适区的人没有人能够在舒适区里取得成功。你刚开始学习的那几秒钟是最困难的。但是如果你能熬过去如果你能熬过刚开始学习的那几秒钟事情就会变得容易很多。你会意识到真正学起来并没有那么困难真正让你心烦的是要去学习这种想法而已。

It's never as bad as you thought it would be.You've got to stop complaining.You've got to stop the negativity .Complaining has zero benefits.It doesn't help anything.It just puts others down with you.You've got to take action in your life.What are you doing about it?You've got to find a solution to your problem.Don't be the problem

事情没你想的那么糟糕你必须停止抱怨你必须停止消极。抱怨对你没有好处，它不会起任何作用它只会拖你的后腿也拖了别人的后腿。你必须动起来。你现在有采取什么行动吗你必须找到解决问题的方法不要让自己成为问题本身。

I see it so many times where someone is getting bad grades but they blame everyone else but themselves.Believe me,that person is going nowhere. Nothing's going to happen in their life.And if that's you,if you're the person that is blaming everyone else for the situation they're in and not taking action for their own responsibilities,then this is for you

有的人他们没能取得好成绩责怪所有人却不反思自己。这样的情况我已经见过太多了。相信我这种人什么都做不成好事不会发生在他们身上。如果你就是这种人如果你不满意现状却要埋怨他人如果你不愿意行动起来为自己负责那么这个演讲就是讲给你听的。

Take action.Take control.If you keep studying at your full potential,and if you keep reading and keep learning and keep asking questions,you'll see the results for yourself.You'll see your grades increase significantly .

采取行动掌握自己的人生。如果你用尽全力去坚持学习如果你坚持阅读坚持学习坚持提问那么你也会看见这带来的结果。你会看见你的成绩有明显的提升。

【译】生命的意义

> 英语学习文章来源　百词斩爱阅读

Edward Perman Cole died in May. it was a Sunday afternoon and there wasn't a cloud in the sky.

爱德华·佩瑞曼·科尔在五月一个周日下午去世了那天天空万里无云。

it's difficult to understand the sum of a person's life. Some people would tell you it's measured by the ones left behind, some believe that it can be measured in faith, some say by love, other folks say life has no meaning at all ...

人生的功过是很难量的。有人说取决于他所留下的事物有人说由信仰来评定也有人说爱能衡量还有人说人生根本没有任何意义　　...

Me, I believe that you measure yourself by the people who measure themselves by you.

而我相信你可以用那些以你为标准的人来衡量自己。

What I can tell you for sure is that by any measure, Edward Cole lived more in his last days on earth than most people manage to wring out of a lifetime. I know that when he died his eyes were closed and his heart was open . . .

而我所能肯定的就是无论你采用什么样的标准爱德华　　·科尔在这世上活着的最后的日子比大多数人穷其一生的日子还要充实。我知道当他去世时他的眼睛是闭上了而他的心灵却是敞开的。

【转】When You Are Old

> 英语学习文章来源　百词斩爱阅读

《当你老了》是威廉·巴特勒·叶芝献给女友毛特·冈妮的真挚爱情诗篇。叶芝是20世纪现代主义诗坛上最著名的诗人他对毛特　·冈妮一见钟情虽多次求婚被拒但仍对她爱慕终身。

原文

```
When you are old and grey and full of sleep,

And nodding by the fire, take down this book,

And slowly read, and dream of the soft look

Your eyes had once, and of their shadows deep;


How many loved your moments of glad grace,

And loved your beauty with love false or true,

But one man loved the pilgrim soul in you,

And loved the sorrows of your changing face;


And bending down beside the glowing bars,

Murmur, a little sadly, how Love fled

And paced upon the mountains overhead

And hid his face amid a crowd of stars.
```

译文一

当你老了头发发白睡意沉沉　　，

倦坐在路边取下这本书来　，

慢慢品读追梦当年的眼神　，

你那柔美的神采与深幽的[illegible]　；

多少人爱过你昙花一现的身影，

爱过你的美貌，以虚伪或真情，

唯独一人曾爱你那朝圣者的心，

爱你衰戚的脸上岁月的痕迹；

在炉栅边，你弯下了腰，

低语着，带着浅浅的伤感，

爱情是怎样逝去，又怎么越过群山，

怎样在繁星之间遮住了脸.

译文二

当汝老去鬓染霜

独伴炉火倦意浅漾

请取此卷曼声吟唱

回思当年汝之[illegible]

眼波深邃[illegible]流光

如花引蝶众生轻狂

彼爱汝貌[illegible]心肠

唯吾一人爱汝心香

知汝心灵圣洁芬芳

当汝老去[illegible]神伤

唯吾一人情意绵长

跪伴炉火私语细量

爱已[illegible]越过[illegible]

爱已[illegible]遁入星光