Undefined && null

Null 为空，undefined为未赋值 Null == undefined ( == 不比较引用地址)

作用域

函数及变量的执行范围 闭包：返回值包含内部函数的引用

引用传递

== && ===

内存释放

新生代、晋升(新生代被回收过，放入老生代)、老生代

ES6

Var & let

Let声明只在块有用，不允许外部访问

Function && =>

= > 里面this，函数生效时的对象

Const && array

Const array，指向array的地址，但array里是可以更改

私有化 && symbol class

Symbol 处理属性名冲突，不能闭包，不能object.keys()访问

Class private方法声明在类外，虽不能访问，但暴露在其他更大的作用域中。

闭包：对外开放非私有部分

Module.exports & exports

AMD, CMD, CommonJS

Require原理

热更新：require cache

上下文VM

**### Module System**

**## Before**

<script1>

<script2>

...

<script9>

**# drawbacks**

global polluted

excute order

difficult management

**## CommonJS**

require('hello');

exports.hello.= () => {}

module.exports = hello;

**# merit**

reuse

**# demerit**

sync load

**# best practise**

Node.js, browersify

**## AMD**

define(id?, dependencies, factory)

define('hello', ['d1', 'd2'], (d1, d2) => {

return ...

})

require(['hello', 'world'], (hello, world) => {

/\* ... \*/

}

**# merit**

Async

**# demerit**

unreadable

**## CMD**

define(function (require, exports, module){

CommonJS

})

**# merit**

Async, lazy excute

**# demerit**

depend on SPM packaged

**## ES6**

babel

**## Module Loading**

Lazy loading: static analysis, deploy&package

**# control module dependencies**

1. optimize denpendencies tree: async(packaged), sync

2. webpack handle JS, loader handle other

3. webpack use Async I/O & multiple cache

**## Middleware**

**# Application-level**

create for app.use()/app.method()

**# Router-level**

express.Route()

**# Error-handle**

handle error

**# Build-in**

express.static

**# Third-party**

**##template engines --- Pug**

Jade --> Pug

app.set('views', './views');

**## Error handling**

**## proxies --- trust proxy**

app.set('trust proxy', '123.123.123.123');

app.set('trust proxy', function(ip) {

if (ip === '127.0.0.1' || ip ==='123.123.123.123') return true

else return false

})

**## Process Managers - PM2**

pm2 start app.js

pm2 list

pm2 stop #id

pm2 restart #id

pm2 show #id

pm2 delete #id

**## Security (SSL, HTTP header, Cookie/Session, dependencies check)**

https://blog.risingstack.com/node-js-security-checklist/

1. package version

2. Use TLS -- SSL

encrypt data before sending to server

Nginx handle TLS

3. Use Helmet

npm i Helmet, set security-related HTTP headers

or

app.disable('x-powered-by');

4. Use cookies securely

express-session: store session id in cookie, and session data in server (Recommend)

cookie-session: serialize session in cookies <=4096

5. Dont use default session cookie name

express-session --> generic cookie name

6. Set cookie security options

secure: ensure from HTTPS

httpOnly: from http/https, not Javascript

domain/path: compare between request

expires: expires date

**## performance & reliablity (DevOps)**

**## gzip, async, logging, handle exception.**

**## node-env == prod, pm2, cluster, cache, load balance, reverse proxy**

**# Dev**

1. gzip (app, nginx)

2. asynchronous functions (node --trace-sync-io)

3. correct logging (sync) --> debug (async), winston

4. handle exception

PM2

try-catch

promise

Error handle in server

not uncaughtexception - domain

**# Ops**

1. Set NODE\_ENV to 'production'

express: cache view template, css, generate less error messge

2. PM2

3. run app in cluster

cluster-service

4. cache request results

Nginx

5. load balancer

Nginx

6. reverse proxy

Nginx