UUID

UUID 是指Universally Unique Identifier，翻译为中文是通用唯一识别码，根据标准方法生成，不依赖中央机构的注册和分配，UUID具有唯一性，这与其他大多数编号方案不同。重复UUID码概率接近零，可以忽略不计。

定义

UUID 是由一组32位数的16进制数字所构成，是故 UUID 理论上的总数为16^32=2^128，约等于3.4 x 10^123。如果用二进制表示，那它就是一个128位的数字。

也就是说若每纳秒产生1百万个 UUID，要花100亿年才会将所有 UUID 用完

格式

UUID使用16进制表示，共有36个字符(32个字母数字+4个连接符”-”)，格式为8-4-4-4-12，如：
6d25a684-9558-11e9-aa94-efccd7a0659b
xxxxxxxx-xxxx-Mxxx-Nxxx-xxxxxxxxxxxx
数字 M的四位表示 UUID 版本(Version)，当前规范有5个版本，M可选值为1, 2, 3, 4, 5；

数字 N的一至三个最高有效位表示 UUID 变体(Variant)。

UUID版本通过M表示，当前规范有5个版本，M可选值为1, 2, 3, 4, 5。这5个版本使用不同算法，利用不同的信息来产生UUID，各版本有各自优势，适用于不同情景。具体使用的信息：

- version 1, date-time & MAC address
- version 2, date-time & MAC address, DCE security version
- version 3, MD5 hash & namespace
- version 4, pseudo-random number
- version 5, SHA-1 hash & namespace

UUID变体通过N表示，当前有4种变体：

- Variant 0, 用一位最高有效位表示0xxx, N = 0……7
- Variant 1, 用两位最高有效位表示10xx, N = 8, 9, a, b （最常用）
- Variant 2, 用三位最高有效位表示110x, N = c, d
- Variant 3, 用三位最高有效为表示111x, N = e, f

使用较多的是版本1和版本4，其中版本1使用当前时间戳和MAC地址信息。版本4使用(伪)随机数信息，128bit中，除去版本确定的4bit和变体确定的2bit(这里用了 Variant 1)，其它122bit全部由(伪)随机数信息确定。

因为时间戳和随机数的唯一性，版本1和版本4总是生成唯一的标识符。若希望对给定的一个字符串总是能生成相同的 UUID，使用版本3或版本5。

随机 UUID 的重复机率

当同一UUID多次生成并分配给不同的引用对象时，就会发生冲突。在使用来自网卡的唯一MAC地址的标准版本1和版本2 UUID的情况下，仅当实现无意或有意地与标准有所不同时，才会发生冲突。

与使用MAC地址生成的版本1和版本2 UUID相比，版本1和-2 UUID使用随机生成的节点ID，基于哈希的版本3和版本5 UUID以及随机的版本4 UUID，即使没有实现问题，也可能发生冲突，尽管发生概率很小，通常可以忽略。该概率可以根据对生日问题的分析来精确计算。

例如，为了产生至少一次碰撞的50%概率而需要生成的随机第4版UUID的数量为2.71亿五千万，计算如下：

$$n \approx \frac{1}{2} + \sqrt{\frac{1}{4} + 2 \times \ln(2) \times 2^{122}} \approx 2.71 \times 10^{18}.$$

这个数字相当于在大约85年的时间内每秒产生10亿个UUID。包含这么多UUID的文件（每个UUID 16个字节）约为45EB = 45000PB = 45000000TB。

通过以下公式近似得出 为找到碰撞概率p而必须生成的最小数量的版本4 UUID：

$$\sqrt{2 \times 2^{122} \times \ln \frac{1}{1-p}}.$$

因此，在103万亿个第4版UUID中找到重复项的可能性是十亿分之一。

前端生产UUID，JavaScript实现

方法 1

最簡單的方法就是使用 Math.random() 產出任意16進位數字，但

- 格式不符合 RFC4122 規範
- Math.random() 產出的數字可能重覆 (collision)，儲存前必須做比對

```
function _uuid() {
  function s4() {
    return Math.floor((1 + Math.random()) *
0x10000).toString(16).substring(1);
  }
  return s4() + s4() + '-' + s4() + '-' + s4() + '-' + s4() + '-' + s4() +
s4() + s4();
}
```

var uuid = _uuid(); //b3165466-df5b-c3d7-0e94-79d94e8c692f
接下來的方法都是基於 Math.random() 做改善，不管是改進格式或唯一性。

方法 2

先擺好格式，再利用 Math.random() 產出任意數字填入格式中。這個方法解決了方法 1 的格式問題，但仍可能有 collision。

```
function _uuid() {
  return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g,
function(c) {var r = Math.random()*16|0,v=c=='x'?r:r&0x3|0x8;return
v.toString(16);});
}
var uuid = _uuid(); //5fc84f46-5743-4ed3-a94d-1ba63b8022a5
```

方法 3

結合 time stamp 與方法 2，同時解決了格式和 collision 問題。

由於加上 Performance.now() (亞毫秒級的時間戳記)，很難產生 collision。主流瀏覽器皆支援，但 IE 只支援 10 以上或 Edge。

```
function _uuid() {
  var d = Date.now();
  if (typeof performance !== 'undefined' && typeof performance.now ===
'function'){
    d += performance.now(); //use high-precision timer if available
  }
  return 'xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx'.replace(/[xy]/g, function
(c) {
    var r = (d + Math.random() * 16) % 16 | 0;
    d = Math.floor(d / 16);
      return (c === 'x' ? r : (r & 0x3 | 0x8)).toString(16);
  });
}
```

測试

```
for(var i = 0; i < 10; i++) {
  console.log(_uuid());
}
```

測试结果

```
b382bd5-becd-4d93-8d3f-98aad78a049e
e3dea0f5-37f2-4d79-ae58-490af3228069
3e8d1c2b-f9f9-4a74-a8af-a4e8bebea438
eefbada8-efee-439a-af83-a80dde423dd6
8084d5dd-4f91-4590-927c-4751b49f2de0
06423d2c-1d62-429b-b6f9-de0a8d4ea95f
```

```
8b86b39e-5f2e-4abd-a8f0-2108a39a1d8a
62c8628e-e2e2-439d-80cd-b86e31e4bd59
1322e5db-91b0-4f12-94bd-fb4989d3cb95
0db3399e-55c3-44ba-822e-afcabf9b702b
```

线上生成 UUID 的网站

http://www.uuid.online/

参考

https://en.wikipedia.org/wiki/Universally_unique_identifier
https://www.jianshu.com/p/da6dae36c290
https://cythilya.github.io/2017/03/12/uuid/