# toggle

# 容器

length 表示一个容器, 'length' 表示字符串, length <- 'length' 表示将字符串放入容器。

# 排序

1. 两两比较排序（冒泡排序）
2. 一个个选择排（选择排序）
3. 起扑克牌法（插入排序）
4. 强迫症收扑克牌法（计数排序）

缺点：

* 排序速度比快排还快；
* 负数和小数不能排序；

1. 快排
2. 归并排序
3. 堆排序
4. 桶排序

将数字分桶，例如1~10一个桶11~20一个桶，每个桶里再排序。

1. 基数排序

先排个位数，后排十位数，再拍百位数。。。。。，先出先进。

# js中的.style只能取内联样式

# 数据结构

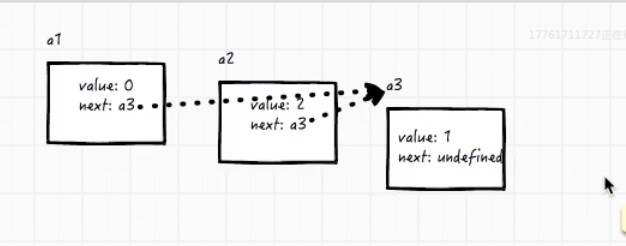
1. 哈希表（Hash Table）
   1. 计数排序中的桶（复杂度 O(n+max)，比快排还快
   2. [桶排序](http://bubkoo.com/2014/01/15/sort-algorithm/bucket-sort/) 与计数排序的区别
   3. [基数排序](http://bubkoo.com/2014/01/15/sort-algorithm/radix-sort/) 与计数排序的区别
2. 队列（Queue）
3. 先进先出
4. 可以用数组实现
5. 举例：排队

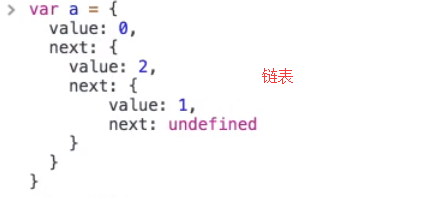
数组.push(“徐涛”)进桶； 数组.shift(“徐涛”)出桶（先进来的先出桶）；

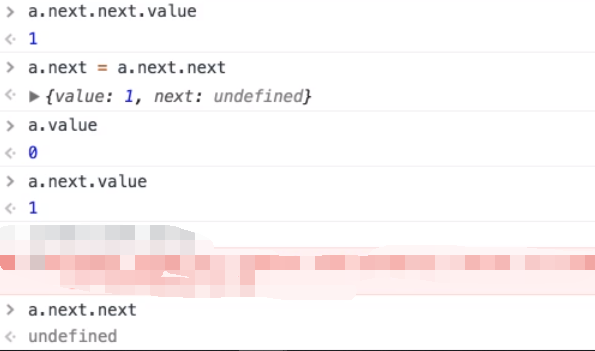
1. 栈（Stack）
2. 先进后出
3. 可以用数组实现
4. 举例：盗梦空间

数组.push(“徐涛”)进栈； 数组.pop(“徐涛”)出栈（从最后开始出栈）；

1. 链表（Linked List）
2. 数组无法**直接**删除中间的一项，链表可以
3. 用哈希（JS里面用对象表示哈希）实现链表
4. head、node 概念







1. 树（tree）
2. 举例：层级结构、DOM
3. 概念：层数、深度、节点个数
4. [二叉树](https://zh.wikipedia.org/wiki/%E4%BA%8C%E5%8F%89%E6%A0%91)
5. 满二叉树
6. 完全二叉树
7. 完全二叉树和满二叉树可以用数组实现
8. 其他树可以用哈希（对象）实现
9. 操作：增删改查
10. 堆排序用到了 tree（和快排差不多）
11. 其他：[B树](https://zh.wikipedia.org/wiki/B%E6%A0%91)、[红黑树](https://zh.wikipedia.org/wiki/%E7%BA%A2%E9%BB%91%E6%A0%91)、[AVL树](https://zh.wikipedia.org/wiki/AVL%E6%A0%91)

# 标识符

1. 第一个字符，可以是任意 Unicode 字母（包括英文字母和其他语言的字母），以及美元符号（$）和下划线（\_）。
2. 第二个字符及后面的字符，除了 Unicode 字母、美元符号和下划线，还可以用数字0-9。

不合法标识符：

1a // 第一个字符不能是数字

23 // 同上

\*\*\* // 标识符不能包含星号

a+b // 标识符不能包含加号

-d // 标识符不能包含减号或连词线

JavaScript有一些保留字，不能用作标识符：arguments、break、case、catch、class、const、continue、debugger、default、delete、do、else、enum、eval、export、extends、false、finally、for、function、if、implements、import、in、instanceof、interface、let、new、null、package、private、protected、public、return、static、super、switch、this、throw、true、try、typeof、var、void、while、with、yield。

# undefined

undefined是一个 JavaScript 关键字，表示“无定义”，如果只是声明变量而没有赋值，则该变量的值；

var a;

a // undefined

# null

null：表示空值，即此处的值为空。

# 条件语句

if结构和switch结构，完成条件判断，即只有满足预设的条件，才会执行相应的语句。

## if语句

if (m === 0) {

// ...

} else if (m === 1) {

// ...

} else if (m === 2) {

// ...

} else {

// ...

}

## switch结构

var x = 1;

switch (x) {

case 1:

console.log('x 等于1');

break;

case 2:

console.log('x 等于2');

break;

default:

console.log('x 等于其他值');

}

## 三元运算符 ?:

(条件) ? 表达式1 : 表达式2

上面代码中，如果“条件”为true，则返回“表达式1”的值，否则返回“表达式2”的值。

var even = (n % 2 === 0) ? true : false;

上面代码中，如果n可以被2整除，则even等于true，否则等于false；

# 循环语句

## while 循环

var i = 0;

while (i < 100) {

console.log('i 当前为：' + i);

i = i + 1;

}

上面的代码将循环100次，直到i等于100为止。

## for 循环

var x = 3;

for (var i = 0; i < x; i++) {

console.log(i);

}

// 0

// 1

// 2

上面代码中，初始化表达式是var i = 0，即初始化一个变量i；测试表达式是i < x，即只要i小于x，就会执行循环；递增表达式是i++，即每次循环结束后，i增大1。

## do…while 循环

do...while循环与while循环类似，唯一的区别就是先运行一次循环体，然后判断循环条件。

do

语句

while (条件);

// 或者

do {

语句

} while (条件);

不管条件是否为真，do...while循环至少运行一次，这是这种结构最大的特点。另外，while语句后面的分号注意不要省略。

下面是一个例子。

var x = 3;

var i = 0;

do {

console.log(i);

i++;

} while(i < x);

## break 语句和 continue 语句

break语句和continue语句都具有跳转作用，可以让代码不按既有的顺序执行。

break语句用于跳出代码块或循环。

var i = 0;

while(i < 100) {

console.log('i 当前为：' + i);

i++;

if (i === 10) break;

}

上面代码只会执行10次循环，一旦i等于10，就会跳出循环。

# 确定值是什么类型（typeof）

JavaScript 有三种方法，可以确定一个值到底是什么类型。

typeof运算符

instanceof运算符

Object.prototype.toString方法

typeof运算符:数值、字符串、布尔值分别返回number、string、boolean;undefined返回undefined; 对象返回object; 空数组（[]）的类型也是object

# 布尔值

空数组（[]）和空对象（{}）对应的布尔值，都是true。

六个值被转为false，其他值都视为true。

* undefined
* null
* false
* 0
* NaN
* ""或''（空字符串）

# 事件

1. onerror

onerror事件会在文档或图像加载过程中发生错误时被触发。

1. previousSibling

previousSibling 属性返回同一树层级中指定节点的前一个节点。