# Math 和 Date

- Math 是 js 的一个内置对象，提供了一堆的方法帮助我们操作 **数字**
- Date 是 js 的一个内置对象，提供了一堆的方法帮助我们操作 **时间**

## Math

- 没有什么多余的东西，就是一堆的方法来操作数字

### random

- `Math.random()` 这个方法是用来生成一个 `0 ~ 1` 之间的随机数
- 每次执行生成的数字都不一样，但是一定是 `0 ~ 1` 之间的
- 生成的数字包含 0 ，但是不包含 1

```
var num = Math.random()
console.log(num) // 得到一个随机数
```

### round

- `Math.round()` 是将一个小数 **四舍五入** 变成一个整数

```
var num = 10.1
console.log(Math.round(num)) // 10

var num2 = 10.6
console.log(Math.round(num2)) // 11
```

### abs

- `Math.abs()` 是返回一个数字的绝对值

```
var num = -10
console.log(math.abs(num)) // 10
```

### ceil

- `Math.ceil()` 是将一个小数 **向上取整** 得到的整数

- `Math.ceil()` 是将一个小数 **向上取整** 得到的整数

```
var num = 10.1
console.log(Math.ceil(num)) // 11

var num2 = 10.9
console.log(Math.ceil(num2)) // 11
```

### floor

- `Math.floor()` 是将一个小数 **向下取整** 的到的整数

```
var num = 10.1
console.log(Math.floor(num)) // 10

var num2 = 10.9
console.log(Math.floor(num2)) // 10
```

### max

- `Math.max()` 得到的是你传入的几个数字之中最大的那个数字

```
console.log(Math.max(1, 2, 3, 4, 5)) // 5
```

### min

- `Math.min()` 得到的是你传入的几个数字之中最小的那个数字

```
console.log(Math.min(1, 2, 3, 4, 5)) // 1
```

### PI

- `Math.PI` 得到的是 `π` 的值，也就是 `3.1415936...`

```
console.log(Math.PI) // 3.141592653589793
```

  - 因为计算机的计算精度问题，只能得到小数点后 15 位
  - **使用 Math.PI 的时候，是不需要加 () 的**

## 数字转换进制

1. `toString()` 方法可以在数字转成字符串的时候给出一个进制数

- 语法： `toString(你要转换的进制)`

```
var num = 100
console.log(num.toString(2)) // 1100100
console.log(num.toString(8)) // 144
console.log(num.toString(16)) // 64
```

2. `parseInt()` 方法可以在字符串转成数字的时候把字符串当成多少进制转成十进制

- 语法： `parseInt(要转换的字符串，当作几进制来转换)`

```
var str = 100
console.log(parseInt(str, 8)) // 64 把 100 当作一个 八进制 的数字转换成 十进制 以后得到的
console.log(parseInt(str, 16)) // 256 把 100 当作 十六进制 的数字转换成 十进制 以后得到的
console.log(parseInt(str, 2)) // 4 把 100 当作 二进制 的数字转换成 十进制 以后得到的
```

# Date

- js 提供的内置构造函数，专门用来获取时间的

## new Date()

- `new Date()` 在不传递参数的情况下是默认返回当前时间

```
var time = new Date()
console.log(time) // 当前时间 Fri Mar 01 2019 13:11:23 GMT+0800 (中国标准时间)
```

- `new Date()` 在传入参数的时候，可以获取到一个你传递进去的时间

```
var time = new Date('2019-03-03 13:11:11')
console.log(time) // Sun Mar 03 2019 13:11:11 GMT+0800 (中国标准时间)
```

- `new Date()` 传递的参数有多种情况

i. 传递两个数字，第一个表示年，第二个表示月份

```
var time = new Date(2019, 00) // 月份从 0 开始计数，0 表示 1月，11 表示 12月
console.log(time) // Tue Jan 01 2019 00:00:00 GMT+0800 (中国标准时间)
```

ii. 传递三个数字，前两个不变，第三个表示该月份的第几天，从 1 到 31

```
var time = new Date(2019, 00, 05)
console.log(time) // Sat Jan 05 2019 00:00:00 GMT+0800 (中国标准时间)
```

iii. 传递四个数字，前三个不变，第四个表示当天的几点，从 0 到 23

```
var time = new Date(2019, 00, 05, 22)
console.log(time) // Sat Jan 05 2019 22:00:00 GMT+0800 (中国标准时间)
```

iv. 传递五个数字，前四个不变，第五个表示的是该小时的多少分钟，从 0 到 59

```
var time = new Date(2019, 00, 05, 22, 33)
console.log(time) // Sat Jan 05 2019 22:33:00 GMT+0800 (中国标准时间)
```

v. 传递六个数字，前五个不变，第六个表示该分钟的多少秒，从 0 到 59

```
var time = new Date(2019, 00, 05, 22, 33, 55)
console.log(time) // Sat Jan 05 2019 22:33:55 GMT+0800 (中国标准时间)
```

vi. 传入字符串的形式

```
console.log(new Date('2019'))
// Tue Jan 01 2019 08:00:00 GMT+0800 (中国标准时间)
console.log(new Date('2019-02'))
// Fri Feb 01 2019 08:00:00 GMT+0800 (中国标准时间)
console.log(new Date('2019-02-03'))
// Sun Feb 03 2019 08:00:00 GMT+0800 (中国标准时间)
console.log(new Date('2019-02-03 13:'))
// Sun Feb 03 2019 13:00:00 GMT+0800 (中国标准时间)
console.log(new Date('2019-02-03 13:13:'))
// Sun Feb 03 2019 13:13:00 GMT+0800 (中国标准时间)
console.log(new Date('2019-02-03 13:13:13'))
// Sun Feb 03 2019 13:13:13 GMT+0800 (中国标准时间)
```

# 将日期字符串格式化成指定内容

- 比如我们得到的时间字符串是 `Sun Feb 03 2019 13:13:13 GMT+0800 (中国标准时间)`
- 我指向得到这个日期中是那一年，我们就要靠截取字符串的形式得到
- 但是现在 js 为我们提供了一系列的方法来得到里面的指定内容

## getFullYear

- `getFullYear()` 方式是得到指定字符串中的哪一年

```
var time = new Date(2019, 03, 03, 08, 00, 22)
console.log(time.getFullYear()) // 2019
```

## getMonth

- `getMonth()` 方法是得到指定字符串中的哪一个月份

```
var time = new Date(2019, 03, 03, 08, 00, 22)
console.log(time.getMonth()) // 3
```

  - 这里要有一个注意的地方
  - 月份是从 0 开始数的
  - 0 表示 1月，1 表示 2月，依此类推

## getDate

- `getDate()` 方法是得到指定字符串中的哪一天

```
var time = new Date(2019, 03, 03, 08, 00, 22)
console.log(time.getDate()) // 3
```

## getHours

- `getHours()` 方法是得到指定字符串中的哪小时

```
var time = new Date(2019, 03, 03, 08, 00, 22)
console.log(time.getHours()) // 8
```

## getMinutes

- `getMinutes()` 方法是得到指定字符串中的哪分钟

```
var time = new Date(2019, 03, 03, 08, 00, 22)
console.log(time.getMinutes()) // 0
```

## getSeconds

- `getSeconds()` 方法是得到指定字符串中的哪秒钟

```
var time = new Date(2019, 03, 03, 08, 00, 22)
console.log(time.getSeconds()) // 22
```

### getDay

- `getDay()` 方法是得到指定字符串当前日期是一周中的第几天（周日是 0，周六是 6）

```
var time = new Date(2019, 03, 08, 08, 00, 22)
console.log(time.getDay()) // 1
```

### getTime

- `getTime()` 方法是得到执行时间到 `格林威治时间` 的毫秒数

```
var time = new Date(2019, 03, 08, 08, 00, 22)
console.log(time.getTime()) // 1554681622000
```

## 获取时间差

- 是指获取两个时间点之间相差的时间
- 在 js 中是不能用时间直接做 减法 的
- 我们需要一些特殊的操作
- 在编程的世界里面，有一个特殊的时间，是 `1970年01月01日00时00分00秒`
- 这个时间我们叫做 `格林威治时间`
- 所有的编程世界里面，这个时间都是一样的，而且 `格林威治时间` 的数字是 0
- 从 `格林威治时间` 开始，每经过1毫秒，数字就会 + 1
- 所以我们可以获取到任意一个时间节点到 `格林威治时间` 的毫秒数
- 然后在用两个毫秒数相减，就能得到两个时间点之间相差的毫秒数
- 我们在通过这个毫秒数得到准确的时间

## 计算时间差

- 例如：我们现在计算一下 `2019-01-01 00:00:00` 到 `2019-01-03 04:55:34` 的时间差

1. 先获取两个时间点到 `格林威治时间` 的毫秒数

```
var time1 = new Date('2019-01-01 00:00:00')
var time2 = new Date('2019-01-03 04:55:34')

time1 = time1.getTime()
time2 = time2.getTime()

console.log(time1) // 1546272000000
console.log(time2) // 1546462534000
```

2. 两个时间相减，得到两个时间点之间相差的毫秒数

```
var differenceTime = time2 - time1
console.log(differenceTime) // 190534000
```

   - 现在我们计算出了两个时间点之间相差的毫秒数

3. 把我们计算的毫秒数换算成时间

   - 先计算出有多少天
   - 以为一天是 `1000 * 60 * 60 * 24` 毫秒
   - 用总的毫秒数除以一天的毫秒数，就能得到多少天了

```
var time1 = new Date('2019-01-01 00:00:00')
var time2 = new Date('2019-01-03 04:55:34')
time1 = time1.getTime()
time2 = time2.getTime()
var differenceTime = time2 - time1

// 计算整的天数
var day = differenceTime / (1000 * 60 * 60 * 24) // 2.20525462962963
day = Math.ceil(day) // 2
```

     - 因为得到的是有小数的天数，我们向下取整，得到有多少个整的天数

   - 使用 `differenceTime` 减去两天所包含的毫秒数，剩下的就是不够一天的毫秒数
   - 用不够一天的毫秒数计算出有多少个小时
   - 因为一个小时是 `1000 * 60 * 60` 毫秒
   - 用不够一天的毫秒数除以一小时的毫秒数，就能得到多少小时了

```
var time1 = new Date('2019-01-01 00:00:00')
var time2 = new Date('2019-01-03 04:55:34')
time1 = time1.getTime()
time2 = time2.getTime()
var differenceTime = time2 - time1

// 计算整的天数
var day = differenceTime / (1000 * 60 * 60 * 24) // 2.20525462962963
day = Math.floor(day) // 2

// 计算整的小时数
```

```
// 计算整的小时数
var afterHours = differenceTime - (1000 * 60 * 60 * 24 * 2)
var hours = afterHours / (1000 * 60 * 60)
hours = Math.floor(hours) // 4
```

- 和刚才一样的道理，我们需要向下取整

- 同理，使用 `afterHours` - 4个小时包含的毫秒数，剩下的就是不够一个小时的毫秒数
- 用不够一个小时的毫秒数计算出有多少分钟
- 因为一分钟是 `1000 * 60` 毫秒
- 用不够一个小时的毫秒数除以一分钟的毫秒数就能得到多少分钟了

```
var time1 = new Date('2019-01-01 00:00:00')
var time2 = new Date('2019-01-03 04:55:34')
time1 = time1.getTime()
time2 = time2.getTime()
var differenceTime = time2 - time1

// 计算整的天数
var day = differenceTime / (1000 * 60 * 60 * 24) // 2.20525462962963
day = Math.floor(day) // 2

// 计算整的小时数
var afterHours = differenceTime - (1000 * 60 * 60 * 24 * 2)
var hours = afterHours / (1000 * 60 * 60)
hours = Math.floor(hours) // 4

// 计算整分钟数
var afterMinutes = afterHours - (1000 * 60 * 60 * 4)
var minutes = afterMinutes / (1000 * 60)
minutes = Math.floor(minutes) // 55
```

- 和之前一样的道理计算出秒

```
var time1 = new Date('2019-01-01 00:00:00')
var time2 = new Date('2019-01-03 04:55:34')
time1 = time1.getTime()
time2 = time2.getTime()
var differenceTime = time2 - time1

// 计算整的天数
```

```
var day = differenceTime / (1000 * 60 * 60 * 24) // 2.20525462962963
day = Math.floor(day) // 2

// 计算整的小时数
var afterHours = differenceTime - (1000 * 60 * 60 * 24 * 2)
var hours = afterHours / (1000 * 60 * 60)
hours = Math.floor(hours) // 4

// 计算整分钟数
var afterMinutes = afterHours - (1000 * 60 * 60 * 4)
var minutes = afterMinutes / (1000 * 60)
minutes = Math.floor(minutes) // 55

// 计算整秒数
var afterSeconds = afterMinutes - (1000 * 60 * 55)
var seconds = afterSeconds / 1000
seconds = Math.floor(seconds) // 34
```

- 最后，同理减去整秒的数，剩下的就是毫秒数

```
var time1 = new Date('2019-01-01 00:00:00')
var time2 = new Date('2019-01-03 04:55:34')
time1 = time1.getTime()
time2 = time2.getTime()
var differenceTime = time2 - time1

// 计算整的天数
var day = differenceTime / (1000 * 60 * 60 * 24) // 2.20525462962963
day = Math.floor(day) // 2

// 计算整的小时数
var afterHours = differenceTime - (1000 * 60 * 60 * 24 * 2)
var hours = afterHours / (1000 * 60 * 60)
hours = Math.floor(hours) // 4

// 计算整分钟数
var afterMinutes = afterHours - (1000 * 60 * 60 * 4)
var minutes = afterMinutes / (1000 * 60)
minutes = Math.floor(minutes) // 55

// 计算整秒数
var afterSeconds = afterMinutes - (1000 * 60 * 55)
```

```
var afterSeconds = afterMinutes - (1000 * 60 * 55)
var seconds = afterSeconds / 1000
seconds = Math.floor(seconds) // 34

// 计算毫秒数
var milliSeconds = afterSeconds - (1000 * 34) // 0
```

- 最后我们把结果输出一下就可以了

```
document.write('2019-01-01 00:00:00 和 2019-01-03 04:55:34 之间相差')
document.write(day + '天' + hours + '小时' + minutes + '分钟' + seconds + '秒' + milliSeconds + '毫秒')
```

# 强化练习1

1. 编写一个函数，能够获得一个随机的 0 ～ 255 之间的数字

```
function getNum() {
 // code in here ...
}

var num = getNum()
// 要求 num 的值在 1 ～ 255 之间（包含 1 和 255）
```

2. 编写一个函数，获得一个十六进制的随机颜色的字符串(例如：#20CD4F)

# 强化练习2

1. 编写函数，把当前时间格式化成 `YYYY-MM-DD HH-mm-ss`
2. 编写函数，要求传入两个时间节点，能返回两个时间节点之间相差 `多少天多少小时多少分钟多少秒`

```
function fn() {
 // code in here ...
}

// console.log(fn(时间节点1， 时间节点2))
// 两个时间节点之间相差 3 天 18 个小时 55 分钟 12 秒
```

# 强化练习3

1. 在页面上出现一个倒计时，显示现在到某一时间点的倒计时（以汉字的形式出现倒计时）
2. 理解复杂数据类型和基本数据类型之间存储的区别
3. 能够使用数组常用方法操作数组
4. 能够使用字符串常用方法操作字符串
5. 能够创建一个对象

# 周四作业

1. 整理本周前三天脑图
2. 将前三天所有练习题一个两遍