# 定时器和延时器

## 定时器

- setInterval()函数可以重复调用一个函数，在每次调用之间具有固定的时间间隔

```
setInterval(function () {
    // 这个函数将自动被以固定间隔时间调用
}, 2000);
```

第一个参数是函数

第二个参数是间隔时间，以毫秒为单位，1000毫秒是1秒

# 函数的参数

- setInterval()函数可以接收第3、4……个参数，它们将按顺序传入函数

```
setInterval(function (a, b) {
    // 形式参数a的值是88，形式参数b的值是66
}, 2000, 88, 66);
```

从第三个参数开始，表示传入函数内的参数

# 具名函数也可以传入setInterval

- 具名函数也可以传入setInterval

```
var a = 0;
function fun() {
    console.log(++a);
}

setInterval(fun, 1000);
```

具名函数当做第一个参数，注意这里没有圆括号！

# 清除定时器

- clearInterval()函数可以清除一个定时器

```
// 设置定时器，并且用timer变量接收这个定时器
var timer = setInterval(function () {

}, 2000);

// 点击按钮时，清除定时器
oBtn.onclick = function () {
    clearInterval(timer);
}
```

用变量接收定时器

清除定时器的时候，要传入定时器变量

# 延时器

- setTimeout()函数可以设置一个延时器，当指定时间到了之后，会执行函数一次，不再重复执行。

```
setTimeout(function () {
    // 这个函数会在2秒后执行一次
}, 2000);
```

# 清除延时器

- clearTimeout()函数可以清除延时器，和clearInterval()非常类似

# 初步认识异步语句

- setInterval()和setTimeout()是两个异步语句

- 异步（asynchronous）：不会阻塞CPU继续执行其他语句，当异步完成时，会执行“回调函数”（callback）

# 初步认识异步语句

异步语句

```
setTimeout(function () {
    console.log('A');
}, 2000);

console.log('B');
```

回调函数

异步语句不会阻塞程序的正常执行

```
B
A
```

# 使用定时器实现动画

# 使用定时器实现动画

- 动画是网页上非常常见的业务需求

# 使用定时器实现动画

- 使用定时器可以实现动画，利用的就是“视觉暂留”原理

# 使用定时器实现动画

- 使用定时器实现动画较为不便：
  ① 不方便根据动画总时间计算步长
  ② 运动方向要设置正负
  ③ 多种运动进行叠加较为困难（比如一个方形一边移动一边变为圆形）