一、BOM 对象

1. BOM 介绍

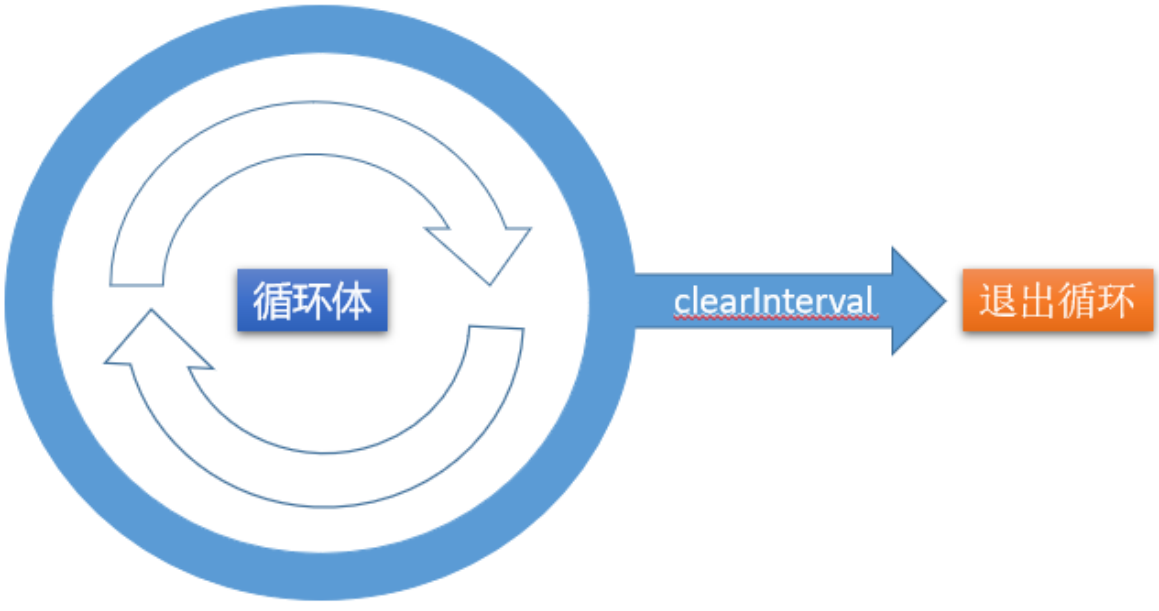
BOM全称为“Browser Object Model”，浏览器对象模型。提供一系列操作浏览器的属性和方法。核心对象为window对象，不需要手动创建，跟随网页运行自动产生，直接使用，在使用时可以省略书写。

2. 对象方法

1. **网页弹框**

```
alert()     //警告框
confirm()   //确认框
```

2. **定时器方法**



周期性定时器 作用：每隔一段时间就执行一次代码

```
//开启定时器:
var timerID = setInterval(function,interval);
/*
参数 :
 function : 需要执行的代码,可以传入函数名;或匿名函数
 interval : 时间间隔,默认以毫秒为单位 1s = 1000ms
返回值 : 返回定时器的ID,用于关闭定时器
*/
```

关闭定时器 :

```
//关闭指定id对应的定时器
clearInterval(timerID);
```

一次性定时器 作用：等待多久之后执行一次代码

```
//开启超时调用:
var timerId = setTimeout(function,timeout);
//关闭超时调用:
clearTimeout(timerId);
```

3. 对象属性

window的大部分属性又是对象类型

1. **history**

   作用：保存当前窗口所访问过的URL 属性 : length 表示当前窗口访问过的URL数量 方法 :

   ```
   back() 对应浏览器窗口的后退按钮，访问前一个记录
   forward() 对应前进按钮，访问记录中的下一个URL
   ```

2. **location**

   作用：保存当前窗口的地址栏信息(URL) 属性 : href 设置或读取当前窗口的地址栏信息 方法 :

   ```
   reload(param) 重载页面(刷新)
   参数为布尔值，默认为 false，表示从缓存中加载，设置为true,强制从服务器根目录加载
   ```

二、DOM节点操作

DOM全称为 “Document Object Model”，文档对象模型，提供操作HTML文档的方法。 （注：每个html文件在浏览器中都视为一篇文档,操作文档实际就是操作页面元素。）

1. 节点对象

JavaScript 会对 html 文档中的元素、属性、文本甚至注释进行封装，称为节点对象，提供相关的属性和方法。

2. 访问节点

- 元素节点 (操作标签)
- 属性节点（操作标签属性）
- 文本节点（操作标签的文本内容）

标签属性都是元素节点对象的属性,可以使用点语法访问，例如：

```
h1.id = "d1";         //set 方法
console.log(h1.id);  //get 方法
h1.id = null;        //remove 方法
```

注意：

- 属性值以字符串表示
- class属性需要更名为 className，避免与关键字冲突，例如： h1.className = "c1 c2 c3";

3. 操作元素样式

1. 为元素添加 id、class属性，对应选择器样式
2. 操作元素的行内样式，访问元素节点的style属性，获取样式对象；样式对象中包含CSS属性，使用点语法操作。

```
p.style.color = "white";
p.style.width = "300px";
p.style.fontSize = "20px";
```

注意：

- 属性值以字符串形式给出，单位不能省略
- 如果css属性名包含连接符，使用JS访问时，一律去掉连接符,改为驼峰， font-size -> fontSize

4. 模拟点击

1. click() 方法可模拟在按钮上的一次鼠标单击。
2. 语法：

   ```
   buttonObject.click();
   ```

   参数 buttonObject 表示按钮元素对象。

三、 实现除重的开奖码生成

1. 页面效果

2. 代码分析

1. **页面元素**

```
<ul class="tmp">
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li></li>
    <li id="btnChange">换一换</li>
</ul>
```

2. **元素样式**

```
ul>li:last-child {
    float: left;
    border: solid 1px #ccc;
    background-color: #eee;
    padding: 5px;
    width: 60px;
    border-radius: 6px;
    margin: 10px 0 5px 10px;
}
```

```
ul>li:not(:last-child) {
    float: left;
    width: 40px;
    background-color: red;
    height: 40px;
    font-size: 20px;
    text-align: center;
    line-height: 40px;
    border-radius: 50%;
    margin: 5px;
    color: #fff;
}
```

3. **创建函数**

```
//编写一个按指范围和位数生成随机数的函数
//max为最大值，len为随机数的位数，blnZ为是否添加0，blnT为是否向上取整
function getRandByParm(max, len, blnZ, blnT) {
    //定义一个数组，保存生成的随机数
    var _arr = [];
    //不断循环检测数组的长度
    while (_arr.length < len) {
        //如果不是向下取整
        if (!blnT) {
            var n = Math.floor(Math.random() * max);
        }
        else { //向上取整
            var n = Math.ceil(Math.random() * max);
        }
        //如果值小于10，则前加上0
        if (n < 10 && blnZ)
            n = "0" + n;
        //定义一个正则表达式
        var reg = new RegExp(n, 'g');
        //验证是否在数组中存在,如果不存，则追加
        if (!reg.test(_arr.toString()))
            //追加到数组中
            _arr.push(n);
    }
    //返回生成好的数组
    return _arr;
}
```

4. **点击事件**

```
var btnChange=document.getElementById("btnChange");
btnChange.onclick=function(){
    var arrTmp=getRandByParm(32,6,true,true)
    var lis=document.getElementsByTagName("li");
    for(var i=0;i<lis.length-1;i++){
        lis[i].innerText=arrTmp[i]
    }
}
```