# javascript基础

> 布尔教育 http://www.itbool.com
> 燕十八 著
> 严禁传播 违者必究

## JS与DOM的关系

浏览器内部有一个js的解释器/执行器/引擎. 如chrome 用V8引擎

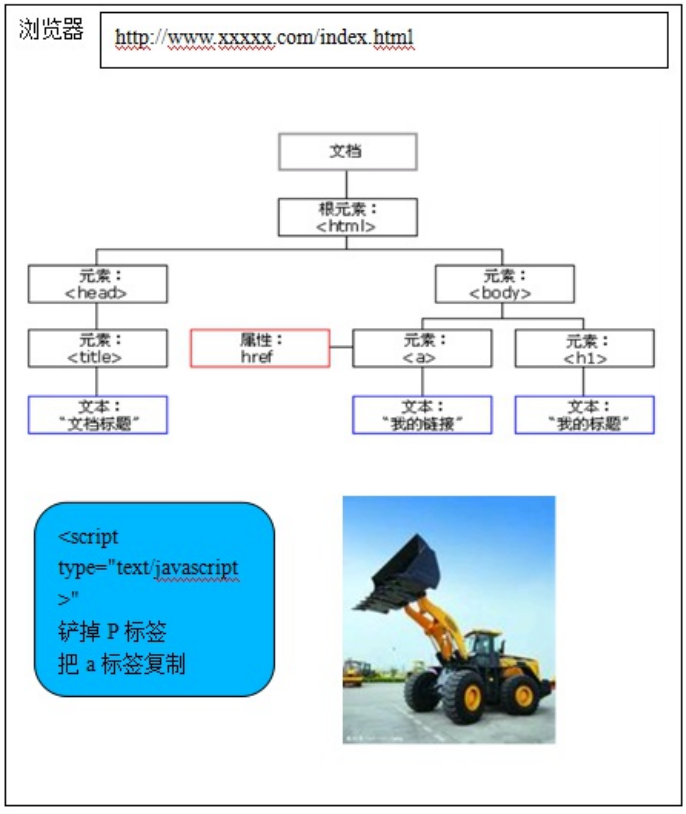
浏览器有渲染html代码的功能, 把html源码在内存里形成一个DOM对象,就是文档对象.

我们在html里写一个js代码, js代码被引擎所执行,而执行的结果,就是对DOM的操作,

而对DOM操作的结果,就是我们常看到的特效,比如,图片漂浮,文字变色.

学习Javascript要分清:
1. js语言本身的语法
2. DOM对象(把body,div,p等节点树看成一个对象)
3. BOM对象 (把浏览器的地址栏,历史记录,DOM等装在一个对象)

浏览器是"宿主",但js的宿主不限于浏览器,也可能是服务器端.
如比较流行的服务器端js框架: node.js




## JS如何引入

1. **页内script代码**

```
<script>
   code........
 </script>
```

1. **外部js文件**

```
<script src="js/jquery.js"></script>
```

**注意:**外部的.js文件里面直接写js代码,
不要在开头和结尾加标签

1. **错误的引入方式**

```
<script src="xx.js">
    var _name = '张三';
    alert(_name);
</script>
```

1. **script标签写在页面的哪个位置?** 页面head和body都可以写.

5.** 多段script的执行顺序?**
按引入顺序,逐段执行.(思考为什么N多页面把JS写在最后?)

# 变量声明

## 命名规范

> JS的变量名 可以用_,数字,字母,$ ,组成,且数字不能开头
> 声明变量 用 var 变量名 来声明

```
var a = 34;
var b = 45;
alert(a+b);

var $ = 'jquery';
alert($);

c = 56;
alert(c);
```

**注意:** 变量名区分大小写 str 和Str不是一个变量
**注意:** 不用var , 会污染全局变量

# JS变量类型

```
// JS变量类型

// 1 数值类型
var a = 23;
var b = 3.14;

// 字符串类型
var c = 'hello';
var d = "world";

// 布尔类型
var e = true;

// null 型
var f = null;

// undefined 型
var g = undefined;

/*
    null是代表对象不存在, 在用DOM操作寻找DOM对象时,没找到,返回null
    如果一个基本型没定义,理解为undefined
*/
```

```
// 数组的写法,"[]"包围,","号隔开每一个值
// JS中,数组是索引类型,key是0,1,2,3,4,5,....
var h = ['张','王','李','赵'];

// 对象(和PHP中的关联数组类似)
var i = {'name':'poly','age':3};

alert(h[2]); // 李
alert(i.age); // 3
alert(i['age']); // 3
```

## 运算符

```
// 运算符
var a = 3;
var b = 2;
alert(a%b); // 1

if(a > b) {
    alert(a + '>' + b);
}

// 在JS中,拼接字符串用"+"

alert('hello' + 3 + 'wolrd'); // hello3world
// 左往右加时,碰到第1个非数值型之后,就理解为字符串拼接
alert(3 + 2 + 'hello' + 5 + 'world'); // 5hello5world,


// 逻辑运算符不同
var c = a||b; // c在PHP中为true, 在JS中,为a的值,3
console.log(c);

a = 0;
b = 9;
c = a||b; // c在PHP中为true, 在JS中,为b的值 ,9
console.log(c);

// 总结,逻辑运算的值,是能确定运算的结果的单元的值
```

## 控制结构

```
// for循环数组
var arr = ['赵','钱','孙','李'];

for(var i=0 ; i<arr.length ; i++) {
    console.log(arr[i]);
}


// 循环对象
var obj = {'name':'lisi','age':29,'height':180};

for(var k in obj) { // 循环obj,把键分别赋给k,
    console.log(k);
    console.log(obj.k); // 错误
    console.log(obj[k]); // 正确
}
```

## 对象操作

```
var str = 'hello world';
console.log(str.length); //11
console.log(str.substr(0,5)); // hello
console.log(str); // hello world;

var arr = ['a','b','c','d'];
console.log(arr.join('~'));
console.log(str.split(' '));
```

- String 字符串对象
  - length属性: 长度
  - concat(String) 连接两个或更多个字符串。
  - indexOf(string) 返回出现字符串的位置
  - substr(num1,[num2]) 截取字符串
  - toLowerCase() 转换成小写
  - toUpperCase() 转换成大写
  - replace(str1,str2) 字符串替换
- Date 日期对象
  - getYear() 返回年份（2位或4位）
  - getFullYear() 返回年份（4位）
  - getMonth() 返回月份 0-11
  - getDate() 返回日期 1-31
  - getDay() 返回星期数 0-6
  - getHours() 返回小时数 0-23
  - getMinutes() 返回分钟数 0-59
  - getSeconds() 返回秒数 0-59
  - getMilliseconds() 返回毫秒数0-999
- Math 数学对象
  - ceil(数值) 大于或等于该数的最小整数
  - floor(数值) 小于或等于该数的最大整数
  - min(数值1,数值2) 返回最小值
  - max(数值1,数值2) 返回最大值
  - pow(数值1,数值2) 返回数值1的数值2次方
  - random() 返回随机数 0---1
  - round(数值) 四舍五入
  - sqrt(数值) 开平方根
- 数组对象
  - concat() 返回一个由两个数组合并组成的新数组。
  - join() 返回一个由数组中的所有元素连接在一起的 String 对象。
  - pop() 删除数组中的最后一个元素并返回该值。
  - push() 向数组中添加新元素，返回数组的新长度。
  - shift() 删除数组中的第一个元素并返回该值。
  - unshift() 返回一个数组，在该数组头部插入了指定的元素。
  - sort() 返回一个元素被排序了的 Array 对象
  - reverse() 返回一个元素反序的 Array 对象。
  - slice() 返回数组的一个片段。
  - splice() 从数组中删除元素，

# 浏览器window对象

**注:** window对象是浏览器宿主对象,和JS语言无关,

**window对象的方法:**
window.alert(message)
window.confirm(message)
window.open(URL , 位置) 打开窗口
window.close() 关闭窗口
widown.print() 打印

window.setIntval(表达式,毫秒);
window.clearIntval(定时器对象);
window.setTimeout(表达式，毫秒)
window.clearTimeout(定时器对象)

**window对象的子对象：**

- navigator 浏览器信息对象
  - appCodeName 内部代码
  - appName 浏览器名称
  - appVersion 浏览器版本
  - platform 操作系统类型
  - userAgent 用户代理信息
  - cookieEnabled 是否支持cookie

判断浏览器是否支持cookie
http://www.zixue.it/thread-12911-1-1.html

- location 地址栏对象
  - host 主机
  - port 端口
  - href 地址
  - pathname 路径
  - protocol 协议
  - search 查询字符串
  - assign(url) 页面跳转
- history 历史记录
  - length : 历史记录的数目
  - back();
  - forward();
  - go();
- screen 屏幕对象
  - height 高度
  - width 宽度
  - availHeight 可用高度
  - availWidth 可用宽度
  - colorDepth 颜色
- document HTML文档对象 即HTML代码形成的对象,操作此对象,可动态的改变页面的内容. 是我们做JS的主战场.

```
// alert是window对象的一个方法,也不是js语言自带的
window.alert('当');
window.confirm('你确认要删除吗?');

window.open('./08-2.html' , '_blank');

// 08-2.html 如下代码:
<input type="button" onclick="window.close()">

// 08-3.html
console.log(window.history);
<input type="button" value="退" onclick="window.history.back()">
<input type="button" value="进" onclick="window.history.forward()">

// 08-4.html
console.log(window.screen.width);
console.log(window.screen.height);
alert(window.document);
```

# 作用域

**var的本质：**

var a = 3; // 声明a变量并赋值
b = 3; // 只是一个"赋值"

**JS作用域的特点:**
首先在函数内部查找变量,找不到则到外层函数查找,逐步找到最外层,
即window对象,并操作window对象的属性.

```
console.log(window.a,window.b);
function t() {
   var a = 'local';
   b = 'global';
}
t();
console.log(window.a,window.b);
```

**思考:** 为什么可以直接写alert(),setInterval()?

# 第一个简单JS效果

```
<style>
#test1{
    width: 300px;
    height: 300px;
    background: blue;
}
#test2 {
    width: 300px;
    height: 300px;
    background: red;
}
</style>
</head>
    <body>
        <div id="test1"  onclick="ch();"></div>
    </body>
    <script>
        function ch() {
            // 1 要找到这个div对象
            var test1 = document.getElementById('test1');
            // 2 要修改它的id属性
            test1.id = 'test2';
        }
    </script>
</html>
```

# 读取和改变对象属性

**小练习:**灯泡开关

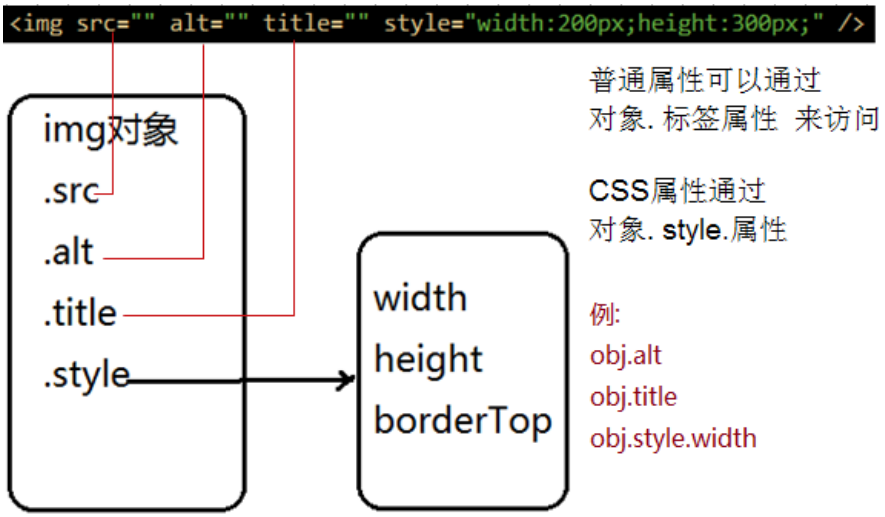

```
<script>
    // 找到对象后,改变其属性,标签的属性,都可以对象.属性 来操作
    // 图片的src属性 <img src="" width="" height="" />

    function qie(){
```

```
        // 查找到对象
        var img = document.getElementsByTagName('img')[0];

        // 修改src
        // img.src = './off.jpg';

        if(img.src.indexOf('off') > -1) {
            img.src = './on.jpg';
        } else {
            img.src = './off.jpg';
        }
    }

    function du() {
        alert(document.getElementsByTagName('img')[0].src);
    }

</script>
</head>
    <body>
        <img src="on.jpg"  onclick="qie();" />
        <input type="button" value="读" onclick="du();" /><br />
    </body>
```

## 找对象

```
<body>
    <h1>关键是找对象</h1>
    <div id="test1">
        <p></p>
        <p></p>
        <p></p>
    </div>
    <div class="test2"></div>
    <input type="text" name="username" id="" value="poly" />
</body>
<script>
    // 按id找
    alert(document.getElementById('test1'));


    // 按标签找,div,p,input,....,哪怕只找到1个,也包装成"数组"来返回的
    var h1s = document.getElementsByTagName('h1');
    alert(h1s);
    alert(h1s[0]);

    // 对于表单元素,可以按name来查询
    alert(document.getElementsByName('username')[0]);
    alert(document.getElementsByName('username')[0].value);

    // 还有,按照类名来找
    alert(document.getElementsByClassName('test2')[0]);

    // 找子对象children,parent,
    alert(document.getElementById('test1').children.length);

    // 变css
    document.getElementsByTagName('h1')[0].style.background='gray';
</script>
```

## 操作对象的属性




**注意:**
标签属性与DOM对象属性的对应关系:
绝大部分2者是相同的,如:
`imgobj.src`属性对应 `<img src="">`中的src属性

但也有例外,如:
`<div class="main">`中,操作class属性用`divobj.className`

css属性与DOM对象属性的对应关系:
2者通过obj.style.css属性名相对应.
如:

```
obj.style.width
obj.style.background
```

如果css属性带有横线,如border-top-style,
则把横线去除并横线后的字母大写.
如:

```
obj.style.borderTopStyle
obj.style.marginLeft
```

**小练习:* *点击div,使其宽,高加10px,底边增宽1px

```
<script>
    // 每点击一次div,宽高增长10px,
    // 同时, 底边的宽加 1 像素
    function bian() {
        // 同学们填写
        var div = document.getElementById('test1');
        var w = parseInt(div.style.width);
        var h = parseInt(div.style.height);
        var b = parseInt(div.style.borderBottomWidth);

        div.style.width = w+10 + 'px';
        div.style.height = h+10 + 'px';
        div.style.borderBottomWidth = b+1+'px';
    }
</script>
<div id="test1" onclick="bian()" style="width:200px;height:200px;border:1px solid blue">
</div>
```

# 获取对象在内存中计算后的样式

上一节中,obj.style只能取得"内联style"的值

对于`<style></style>`中的css属性值,则无能为力.

我们可以用`obj.currentStyle`,和`window.getComputedStyle()`来获取
。
注意：只有 IE 和 Opera 支持使用 currentStyle 获取 HTMLElement 的计算后的样式，其他浏览器中不支持。
标准浏览器中使用getComputedStyle，IE9及以上也支持getComputedStyle。

window.getComputedStyle(obj,伪元素)
参数说明:
第一个参数为要获取计算后的样式的目标元素
第二个参数为期望的伪元素，如 ':after'，':first-letter' 等,一般设为null

考虑兼容性,封装函数

```
function getStyle (el,attr){
    return el.currentStyle?el.currentStyle[attr]:getComputedStyle(el,null)[attr];
}
```

**注意:** 这2个方法,获取的对象是*只读*的,要改样式,还得靠obj.style

# 删除对象

**步骤:**
1. 找到对象
2. 找到他的父对象parentObj
3. parentObj.removeChild(子对象);

**练习:* 删除最后一个li

```
<ul>
 <li>春</li>
 <li>夏</li>
 <li>秋</li>
 <li>冬</li>
</ul>
```

# 创建对象

**步骤:**
1. 创建对象
2. 找到父对象parentObj
3. parentObj.addChild(对象); appendChild

练习: 添加1个li

```
<ul>
 <li>春</li>
 <li>夏</li>
 <li>秋</li>
</ul>
```

```
function dong() {
    // 创建文本节点
    var txt = document.createTextNode('冬');

    // 创建li
    var li = document.createElement('li');
    li.appendChild(txt);

    // 获取ul
    var ul = document.getElementsByTagName('ul')[0];
```

```
    // 添加子元素
    ul.appendChild(li);
}
```

## 暴力操作节点

**innerHTML:** 代表节点内的内容,能读能写
不是一个w3c规定的标准对象属性,但是---各浏览器支持的很好

```
<script>
// 简单粗暴的插入节点的方式
function tian() {
    // 获取父节点
    var main = document.getElementById('main');

    // 拼接html代码
    var html = '<ul><li>春</li><li>夏</li><li>秋</li></ul>';

    // 一次插入
    main.innerHTML = html;
}

function tian2() {
    var ul = document.getElementsByTagName('ul')[0];
    var winter = '<li>冬</li>';
    ul.innerHTML += winter;
}
</script>
<input type="button" value="添加春夏秋" onclick="tian();" />
<input type="button" value="添加冬" onclick="tian2();" />
<div id="main"></div>
```

## 联动菜单

所用知识点: 事件+DOM操作

```
<script>
    var area = [ ['朝阳' , '海淀' , '昌平'] , ['淮北','淮南' , '铜陵'] ];
    /**
    1: 选择select时,如何触发? 哪种事件? onchange
    2: 如何获取被选中的option的值 ? selectobj.value
    3: 动态生成option innerHTML
    **/

    function ld() {
        var psel = document.getElementsByName('pro')[0];
        //alert(psel.value);

        if(psel.value == '') {
            return;
        }

        for(var i=0 ,str='', len=area[psel.value].length; i<len; i++) {
            str = str + '<option value="' + i + '">' + area[psel.value][i] + '</option>';
        }

        document.getElementsByName('city')[0].innerHTML = str;

    }

</script>
<select name="pro" onchange="ld()">
    <option value="">请选择</option>
    <option value="0">北京</option>
```

```
    <option value="1">安徽</option>
</select>
<select name="city">
</select>
```

# 定时器

window.setTimeout('语句',毫秒); 指定毫秒后执行一次语句
注:定时器不属于JS的知识,他是window对象提供的功能

```
function bom() {
    var img = document.getElementsByTagName('img')[0];
    img.src = 'warn.jpg'; // 3秒后改变图片的src
}

window.setTimeout('bom()' , 3000);

</script>
<h1>定时器</h1>
<img src="./bom.jpg" alt="" />
```

window.setInterval('语句' , 毫秒); // 每经过N毫秒执行语句

**小练习**: 倒计时炸弹
思路: 每1秒,修改剩余时间,剩余时间为0时,修改爆炸后的图片.

```
<script>
    function t() {
        var inp = document.getElementsByName('time')[0];
        var t = parseInt(inp.value);

        inp.value = --t;

        if(t == 0) {
            document.getElementsByTagName('img')[0].src = './warn.jpg';
        }
    }

    setInterval('t()' , 1000);
</script>
<body>
    <p>
        <input type="text" name="time" value="5" />
    </p>
    <img src="bom.jpg" alt="" />
</body>
```

**清除定时器:**

- clearInterval()
- clearTimeout()

例:改进定时炸弹,爆炸后不再倒计时

```
function t() {
    var inp = document.getElementsByName('time')[0];
    var t = parseInt(inp.value);

    inp.value = --t;

    if(t == 0) {
        document.getElementsByTagName('img')[0].src = './warn.jpg';
        clearInterval(clock);
    }
}

var clock = setInterval('t()' , 1000);
```

**setTimeout实现倒计时**
setTimeout可以反复调用自身,达到类似setInterval的效果
例:setTimeout版的定时炸弹

```
function t() {
    var inp = document.getElementsByName('time')[0];
    var t = parseInt(inp.value);

    inp.value = --t;

    if(t == 0) {
        document.getElementsByTagName('img')[0].src = './warn.jpg';
    } else {
        setTimeout('t()' , 1000);
    }
}

setTimeout('t()' , 1000);
```

# 事件

## 常用事件

- onclick 元素点击时
- onfocus 元素获得焦点时
- onblur 元素失去焦点时

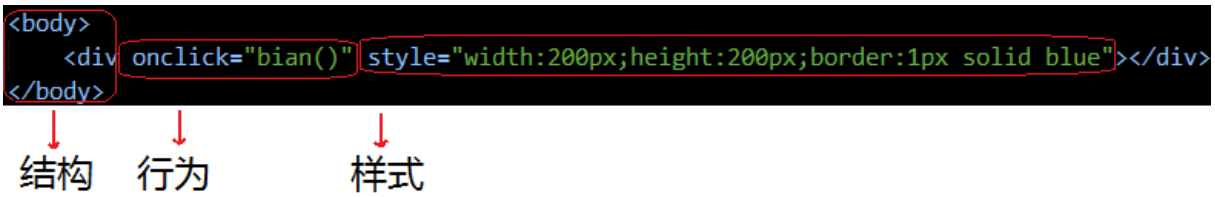
```
<form action="">
    <p>用户名:<input type="text" onfocus="f1();" onblur="f2()" /></p>
    <p>Email: <input type="text" /></p>
    <p>密码:<input type="text" /></p>
</form>
<script>
function f1() {
    document.getElementsByTagName('input')[0].style.border = '1px solid gold';
}

function f2() {
    var inp = document.getElementsByTagName('input')[0];
    if(inp.value == '') {
        inp.style.border = '1px solid red';
        alert('请填写用户名');
        setTimeout(function(){inp.focus();} , 1);
    }
```

```
}
</script>
```

**注意:** 在失去焦点的函数中,直接调用得到focus()方法,在ff中不起作用.
**参考:** http://stackoverflow.com/questions/4640687/javascript-onchange-onblur-and-focus-weirdness-in-firefox

- onmouseover 鼠标经过时
- onsubmit 表单提交时
- onload 页面加载完毕时 注意: onsubmit="return func()"; func()函数才能拦截提交的效果
- oninput input内容变化时触发[HTML5新增的,IE9及以上才能用]

## 结构样式行为相分离




## 事件对象

**事件对象:**事件发生的瞬间,发生位置,时间,鼠标按键,触发的节点等信息,被打包成1个对象.
此对象,系统自动传递给事件函数的第1个参数.

**小练习:**抓不到的美女

```
<style>
    img {
        display: block;
        width: 130px;
        height: 130px;
        position: relative;
    }
</style>
<body>
    <img src="./mn.png" alt="" />
</body>
<script>
var img = document.getElementsByTagName('img')[0]
img.onmouseover=function (ev) {
        //var img = document.getElementsByTagName('img')[0];

        img.style.left = ev.pageX + 130 + 'px';
        img.style.top = ev.pageY + 130 + 'px';
    }
</script>
```

## 事件委托

当有比较多的元素需要绑定某事件时,可以把事件绑定在他们的父元素上,
委托给父元素来处理.

```
<table>
    <tr><td></td><td></td></tr>
    <tr><td></td><td></td></tr>
</table>
<script>
    document.getElementsByTagName('table')[0].onclick = function(ev) {
        ev.target = ev.target || ev.srcElement;

        ev.target.style.backgroundColor='black';
    }
</script>
```

# JS使用正则

### 声明

```
var patt = /^\d{11}$/;
```

### 使用

// 判断String是否符合正则要求
patt.test(String);

// 找出字符串中的符合正则的子串
patt.exec(String);

例:表单验证

```
<form action="" onsubmit="return check()">
    <p>用户名:<input type="text" name="username" />(数字+字母,6-16位)</p>
    <p>Email:<input type="text" /></p>
    <p><input type="submit" value="注册" /></p>
</form>
<script>
    function check() {
        var inps = document.getElementsByTagName('input');

        var patt = /^[a-zA-Z0-9]{6,16}$/;
        if(!patt.test(inps[0].value)) {
            alert('请输入正确的用户名');
            return false;
        }

        var patt = /^\w+@[a-zA-Z0-9\-]+(\.[a-zA-Z0-9\-]+)\1*$/;
        if(!patt.test(inps[1].value)) {
            alert('请输入正确的email');
            return false;
        }
    }
</script>
```

# 面向对象基础

### json对象

js允许通过**键值对**形式写对象.
如`var obj = {name:'lisi',age:29}`

此结构可以**嵌套**书写,如:

```
var book = {
title:'天龙八部',
author:'金庸',
role:{
    name:'虚竹',
    job:'和尚'
    }
}
```

键值也可以是**方法**,如:

```
var dog = {
```

```
    name : 'tom';
    bark: function() {
        alert('I am' + this.name);
    }
}

dog.bark(); // I am tom
```

这种键值对结构,在**其他语言**中,也有类似结构.
如PHP中的关联数组`array('name'=>'lisi' , 'age'=>29)`
python中的字典`{'name':'lisi' , 'age':29}`

因此,互联网上很多API交换数据时,通过JSON键值对的格式来传输.
即大家都把数据转成JS的json对象格式,以此格式做标准来交流.

如PHP中,可以用json_encode,js_decode来编码和解码.

```
$obj = array(
'name'=>'lisi',
'age'=>29
    );

// PHP数组编成json格式
echo json_encode($obj); // {"name":"lisi","age":29}

// json格式解码成PHP数组形式
$str = '{"name":"lisi","age":29}';
print_r(json_encode($str,true)); // array('name'=>'lisi','age'=>29);
```

## 自己写构造函数

```
// js中没有类,只有构造方法
function Dog(name,color) {
    this.name = name;
    this.color = color;
    this.bark = function() {
        alert('my name is ' + this.name + ',color:' + this.color );
    }
}

var cat = {
    name:'kitty',
    color:'yellow',
    climb: function() {
        alert('I am ' + this.name);
    }
}

var dog = new Dog('xiaohei' , 'black');
console.log(dog);

dog.bark();

/*
如果直接运行构造方法,Dog(xx,xx);
this是谁? this是null,在目前的js标准中,this为null时,会把this指向window
发导致window上多了name,color,bark这几个全局变量
*/
/*
// new的过程中发生了什么?
先造一个空对象{}
this指向该空对象
执行函数体: 对象.name = 传来的值 对象的color = 传来的值
返回该对象
*/
```

### 系统自带对象

```
var dt = new Date();
console.log(dt.getFullYear());

// Math 已经是一个现成的对象
//alert(Math.random());
alert(Math.floor(Math.random()*6+5));
```

### 自动装箱的对象

```
var str = 'hello';
str.substr();
```

### this是谁

```
// js中,this始终指向当前调用者对象

window.name = 'chrome';

function t() {
    alert(this.name);
}

var dog = {name:'xiaohei'};
dog.bark = t;

dog.bark(); // xiaohei


var cat = {name:'kitty'}
cat.bark = dog.bark;
cat.bark(); // kitty

var tmp = cat.bark;
tmp(); // chrome

//
(cat.bark=cat.bark)();
```

## 别踩白块

### 第1步 框架搭建

```
<style>
.row div {
    width: 98px;
    height: 98px;
    border: 1px solid gray;
    float:left;
}

.row {
    width: 400px;
    height: 100px;
}

.black {
    background: black;
}
```

```
#main {
    position: relative;
    top: -100px;
}

#cont {
    position: relative;
    border: 1px solid red;
    width: 400px;
    overflow: hidden;
}
</style>
<div id="cont">
    <div id="main">
        <div class="row">
            <div></div>
            <div></div>
            <div class="black"></div>
            <div></div>
        </div>
        <div class="row">
            <div></div>
            <div></div>
            <div></div>
            <div></div>
        </div>
        <div class="row">
            <div></div>
            <div></div>
            <div></div>
            <div></div>
        </div>
        <div class="row">
            <div></div>
            <div></div>
            <div></div>
            <div></div>
        </div>
    </div>
</div
```

### 第2步 动态创建div

```
var main = document.getElementById('main');
/**
* 创建单个的div对象
* @param className 类名称
*/
function cdiv(className) {
    var div = document.createElement('div');
    if(className) {
        div.className = className;
    }

    return div;
}

/**
* 创建一行
*/
function crow() {
    var row = cdiv('row');
    var index = Math.floor(Math.random()*4);

    for(var i=0; i<4; i+=1) {
        if(i==index) {
```

```
            row.appendChild(cdiv('black'));
        } else {
            row.appendChild(cdiv());
        }
    }

    //return row;
    if(main.firstChild) {
        main.insertBefore(row , main.firstChild);
    } else {
        main.appendChild(row);
    }
}

// 初始化出4行来
function init() {
    for(var i=0; i<4; i+=1) {
        crow();
    }
}

init();
```

### 第3步 白块动起来

```
var clock = null;

// 4行黑白块整体下移
function move() {
    var top = parseInt(getStyle(main , 'top'));
    top += 2;

    main.style.top = top + 'px';
}

// 初始化出4行来
function init() {
    for(var i=0; i<4; i+=1) {
        crow();
    }

    clock = setInterval('move()' , 40);
}
```

### 第4步 div块循环出现

```
// div循环出现
function move() {
    var top = parseInt(getStyle(main , 'top'));
    top += 2;

    main.style.top = top + 'px';

    if(top == 0) {
        crow();
        main.style.top = '-100px';

        if(main.children.length>5) {
            main.removeChild(main.lastChild);
        }
    }
}
```

### 第5步 加分

```
function init() {
    for(var i=0; i<4; i+=1) {
        crow();
    }

    main.onclick = function(ev) {
        if(ev.target.className=='') {
            alert('输了');
        } else {
            score();
        }
    }

    clock = setInterval('move()' , 40);
}

function score() {
    var h2 = document.getElementsByTagName('h2')[0];
    var s = parseInt(h2.innerHTML)+1;
    h2.innerHTML = s;
}
```

### 第6步 判断输赢

```
var state = true;
// 判断黑白块,加分或者输掉
function init() {
    for(var i=0; i<4; i+=1) {
        crow();
    }

    main.onclick = function(ev) {
        if(state == false) {
            alert('下次再玩吧');
            return;
        }

        if(obj.className == '') {
            fail();
        } else {
            score(ev.target);
        }
    }

    clock = setInterval('move()' , 40);
}

// 输掉
function fail() {
    clearInterval(clock);
    state = false;
    alert('输了');
}

function score(obj) {
    var h2 = document.getElementsByTagName('h2')[0];
    var s = parseInt(h2.innerHTML)+1;
    h2.innerHTML = s;

    obj.className = '';
    obj.parentNode.pass = true;
}
```

## 第7步 加速

```
var speed = 5;
// 加分函数
function score() {
    var h2 = document.getElementsByTagName('h2')[0];
    var s = parseInt(h2.innerHTML)+1;
    h2.innerHTML = s;
    if(s % 10 == 0) {
        speed += 2;
    }
}

// 白块移动函数
function move() {
    var t = parseInt(getStyle(main , 'top')) + speed;// 注意getStyle函数是第14节课程时封装的
    main.style.top = t + 'px';

    if(t >= 0) {
        crow();
        main.style.top = -100 + 'px';

        // 判断最后一行是否过关
        if(main.lastChild.pass == false) {
                fail();
        }

        // 如果多于5行,则删掉
        if(main.children.length > 5 ) {
            main.removeChild(main.lastChild);
        }
    }

}
```