==========

更多干货技术访问小D课堂官网 https://xdclass.net

小D课堂QQ客服Vicky：561509994

公众号搜索：小D课堂

==========

愿景："让编程不在难学，让技术与生活更加有趣"

第一章 欢迎走进javascript的世界

第一节 走进JS的世界，探索JS的魅力

- 我们为什么要学习javascript?
- 零基础可以学习javascript吗?
- 为什么我要选择这个课程学习?

第二节 JavaScript组成成员讲解

- **JS到底是什么?**
  - **JavaScript一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型**
  - **Javascript是当今最流行的脚本语言，我们生活中看到的网页和基于html5的app里面的交互逻辑都是由javascript驱动的**
  - **一句话概括就是javascript是一种运行在浏览器中的解释型的编程语言**

- JS由什么组成的呢?
    - ECMAScript : 解释器、翻译者 (描述了该语言的语法和基本对象) 它是javascript的标准
    - DOM : Document Object Model 文档对象模型 W3C是DOM的标准
    - BOM : Browser Object Model 浏览器对象模型 缺乏标准

第三节 开始第一个JS脚本的编写

- 确定我们要做的一个效果是鼠标移入div背景颜色变成红色，鼠标移出，背景颜色变回白色
    - 第一步：我们要知道鼠标移入事件和鼠标移出事件
        - 鼠标移入事件 onmouseover
        - 鼠标移出事件 onmouseout
    - 第二步：我们要找到需要改变的div元素
        - 我们通过id就可以找到我们要改变的div
    - 第三步：我们要对修改我们的css属性
        - js操作是跟我们的html标签的属性是一一对应的(只有一个特例class，因为class是js的关键字，是类的意思)
        - 计算机里面识别我们js里面 . 就是的的意思

愿景："让编程不在难学，让技术与生活更加有趣"

第二章 初始javascript

第一节 实战必备技能获取标签元素并进行操作

- **document.getElementById('ID名') 返回这个id名的元素**
- **document.getElementsByTagName('标签名') 返回所有这个标签名的元素的集合**
- **document.getElementsByClassName('class名') 返回所有这个class的元素的集合**
- **document.querySelector("css任意选择器") 返回第一个**
- **document.querySelectorAll("css任意选择器") 返回符合的所有**
- **document.documentElement 获取HTML元素**
- **document.body 获取body元素**

第二节 深入了解JS编写的最佳位置

- **一般情况下JS是写在页面的任何位置都可以的，需要script标签包着，但是别写在html标签外**
- **最常见的是写在head标签内部和body内部**
- **写在head标签内需要写上window.onload包着，要不然会报错，因为执行js时页面标签还没加载**
    - **window.onload的作用是当页面加载完成后自动触发事件**
    - **一个页面应该只有一个window.onload事件，因为如果写多个的话会覆盖只执行最后一个**
- **总结：js理应写在body结束标签之前**

第三节 详细分析JS输出及调试方式

- 弹窗型输出
  - alert('输出内容')
- 浏览器调试窗口输出
  - console.log('输出内容')
- innerHTML和innerText
  - 给获取到的元素一些内容
  - document.getElementById('id名').innerHTML='内容'
- document.write('输出内容')
  - 输出内容会清空原有的html再生成一个新的html
- 注意，我们的document.write()重写文档是发生在文档加载完成后事件触发里面

第四节 演示何为JS事件及演示常见的事件

- JS事件就是我们的行为能被侦测到，且触发相对应的函数(函数里面写我们要做的事情)
- 常见的事件有
  - onclick 鼠标点击事件
  - onmouseover 鼠标被移到某元素之上
  - onmouseout 鼠标从某元素上面移开
  - onchange 元素值改变，一般用在表单元素上
  - onkeydown 用户按下键盘按键
  - onfocus 元素获得焦点
  - onblur 元素失去焦点
  - window.onload 页面加载完成

第五节 动手操作DIV的任意样式

- 先创建一个div获取到div元素，然后对其进行操作
    1. 创建div 和操作按钮
    2. 获取div和按钮
    3. 给按钮添加点击事件
    4. 操作div样式

第六节 深度讲解JS的数据类型

- 基础数据类型
    - 字符串 string
    - 数字(整型浮点型) number
    - 布尔 boolean
    - null 空对象
    - undefined 未定义
- 复杂数据类型
    - 数组 Array
    - 对象 Object
- 检测数据类型的两种基本方法
    - typeof
    - Object.prototype.toString.call('数据') //鉴别复杂数据类型、引用数据类型

第七节 解释变量出现的意义

- 如何定义变量(创建变量)

    - **用var来定于变量 var name**
- **变量名怎么取，有什么限制**
    - **变量的首字母必须是由字母（a-zA-Z）或下划线（_）或美元符（$）开头,不能是数字**
    - **后面的可以是字母（a-zA-Z）或下划线（_）或美元符（$）或者是数字，并且是区分大小写的**
    - **区分大小写的意思是 var name 和var Name是定义两个不同的变量**
- **变量也可以理解为取一个别名，方便操作**

第八节 数据类型之间的转换

- 显式转换(强制类型转换)
    - Number()
    - parseInt()
    - parseFloat()
    - String()
    - toString()
    - Boolean()
- 隐式转换
    - 操作符或者语句进行运算时
    - if while比较里面也会进行隐式转换

第九节 讲解如何创建复杂数据类型数组和对象

- **数组创建**
    - **直接创建**

        - var arr=[] //创建空数组
        - var arr1=[1,2] //创建有内容的数组
    - 利用构造函数创建
        - var arr1 = new Array();//创建空数组
        - var arr2 = new Array(10)；//创建一个长度为10的数组
        - var arr3 = new Array(5,4,3,2,1)；//创建数组并初始化
- 对象创建
    - 直接创建
        - var obj={} //创建空对象
        - var obj1={a:1,b:2} //创建有内容的对象
    - 利用构造函数创建
        - var obj=new Object()

愿景："让编程不在难学，让技术与生活更加有趣"

第三章 初识函数及作用域

第一节 深度讲解函数出现的意义及其定义与调用

- 函数出现的意义
    - 方便我们封装、调用代码
- 函数的定义
    - function fun(){console.log(1)}
    - var fun=function(){console.log(2)}

    - var fun= new Function(console.log(3)) //这种方式函数会自调用(函数自己完成调用动作)
- 函数的调用
    - fun()

第二节 快速了解JS命名规范

- 变量 小驼峰命名法
    - 第一个单词以小写字母开始，从第二个单词开始以后的每个单词的首字母都采用大写字母
- 函数名 跟变量一样
- 常量 全大写单词间用下划线隔开(_)

第三节 实战中函数的几种写法讲解

- function fun(){} //最常见的声明
- (function fun(){}) //函数表达式声明方式，只在括号内起作用，外部无法访问
- (function(){}) //匿名函数表达式，外部无法访问
- new Function() //通过构造器方式，因为麻烦，很少用
- export default function(){} //es6的向外暴露方法
- ()=>{} //箭头函数，在这里大家知道箭头函数就可以了，后续es6会详细讲解

第四节 实战必备技能之函数的传参及取参

- 函数里的传参取参相当于在函数内部放一个占位符，这个占位符的内容由外部调用的时候传入

第五节 函数传参取参之不定参arguments讲解

- 场景：我需要写一个方法来做求和，但是我不知道我传入的参数有多少个
  - 例如函数名是sum 我调用的时候可能是sum(1,2,3,4,5)也有可能是sum(1,2,3,4
  - 这个时候该怎么编写这个求和函数呢?

第六节 深度剖析变量作用域与闭包

- 变量分为局部变量和全局变量
  - 局部变量 只在定义的函数内部使用
  - 全局变量 可以在整个script作用域内都可以使用
- 作用域链，只有在自己当前的小作用域内找不到才会向父级作用域寻找，直到找不到为止(存在同名变量时，在当前作用域内局部变量会覆盖全局变量，局部变量优先级比较高)
- 闭包简单理解：可以调用函数内部变量的函数

第七节 讲解函数的返回值

- 返回值是函数处理完返回的一个结果，用return 返回
- 一个函数内部可以有多个return，但是return代表着函数结束，一执行return函数就会结束运行
- 不写return函数默认返回undefined
- 想返回多个值可以把多个值封装成数组或json对象

愿景："让编程不在难学，让技术与生活更加有趣"

第四章 javascript程序的流程控制

第一节 讲解实战知识之JS判断

- if、else if、else
  - If(判断条件){执行的操作} 关键字if是固定写法 括号()里面写判断条件 花括号{}里面写操作
- 三元运算符（也叫三目运算）
  - 条件? '成立':'不成立'
- switch case 比较高级的写法，一般是判断选择条件比较多时使用，属于全等判断
  - 每一个case都需要写上break 要不然就下面的条件不成立也会执行

第二节 讲解实战知识之JS循环

- **了解JS循环的操作流程**

  1、声明循环变量； 2、判断循环条件；3、执行循环体操作； 4、更新循环变量； 5、然后循环执行2-4，直到条件不成立，跳出循环。

- **while**
  - **while循环特点：先判断后执行；**

```
var tim = 1;//1、声明循环变量

while (tim<=10){//2、判断循环条件;
   console.log(tim);//3、执行循环体操作;
    tim++;//4、更新循环变量;
}
```

- **do while**
  - **do-while循环特点：先执行再判断，即使初始条件不成立，do-while循环至少执行一次；**

```
var tim = 10;

do{
    console.log(tim);//10 9 8 7 6 5 4 3 2 1 0
    tim--;
    }while(tim>=0);

console.log(tim);//-1
```

- **for**
  - **for有三个表达式：①声明循环变量；②判断循环条件；③更新**

循环变量；
    - for循环的执行特点：先判断再执行，与while相同
    - 循环数组
- for in
    - 循环遍历对象
    - for in中的格式：for(keys in zhangsan){} keys表示obj对象的每一个键值对的键，所有循环中，需要使用obj[keys]来取到每一个值！！！

第三节 流程控制X因素break continue讲解

- break 一般用于跳出整个循环，循环体后面的语句会继续执行
- continue 语句中断循环中的迭代，如果出现了指定的条件，然后继续循环中的下一个迭代

第四节 JS判断里什么是真什么是假

- 真: 除了假都是真
- 假: false 数字0 空字符串 null undefined NAN

第五节 JS逻辑运算符全方位讲解

- 逻辑运算符有 与(&&) 或(||) 非(!)

- && 全部真才返回真
- || 一个真就返回真
- ！ 真变成假，假变成真

愿景："让编程不在难学，让技术与生活更加有趣"

第五章 深入了解JSON

第一节 深入了解JSON到底是什么

- **JSON是JavaScript Object Notation的缩写，它是一种数据交换格式**
- **JSON基于文本，优于轻量，用于交换数据**
- **JSON主要用于前后端交互时数据的传输，JSON简单易用，是ECMA的一个标准，几乎所有的编程语言都有解析JSON的库，所以我们可以放心大胆的在前后端交互里直接使用JSON**
- **JSON跟XML的比较**
  - **在可读性上面JSON是简易的写法，XML是规范的标签写法形式**
  - **使用层面，由于JSON是我们ECMA的一个标准，所以在我们JS上面使用有着天然的主场优势，对于存储我们JS的各种格式的数据都比XML有优势**
  - **轻量级是JSON最大的优势，在数据交换这一块轻量就代表着速度要比别人快**

第二节 深度剖析JSON字符串、JSON对象和数组

- **JSON的语法规定JSON字符串必须为双引号("")包着而不是单引号('')，同时JSON对象内的键也是必须为双引号("")包着而不是单引号('')**
- **JSON对象**

```
{"name":"JSON对象","address":"广东省广州
市","age":28}
```

- **JSON数组**

```
[
  {"name":"JSON数组1","address":"广东省广州
市","age":28},
  {"name":"JSON数组2","address":"广东省广州
市","age":28}
]
```

- **JSON字符串**

```
'{"name":"JSON对象","address":"广东省广州
市","age":28}'
```

第三节 实战必备技能之JSON.parse()和JSON.stringify()的使用

- 序列化 JSON.stringify()
  - 把js对象转JSON字符串
- 反序列化 JSON.parse()
  - 把 JSON字符串转js对象

第四节 详细讲解JSON的使用

- 对于JSON对象的操作
  - 增 obj.xxx='xxx'
  - 删 delete obj.xxx
  - 改 obj.xxx='xxx'
  - 查 obj.xxx
- 对获取到的数组进行简单的操作
  - 模拟获取到后端的数组然后循环输出

第五节 实战必备技能之数组常见操作讲解(一)

- shift() 删除数组第一个元素
  - 删除数组第一个元素 返回被删除元素
- pop() 删除数组最后一个元素
  - 删除数组最后一个元素 返回被删除元素
- unshiift() 向数组第一个元素前面添加一个元素
  - 向数组第一个元素前面添加一个元素 返回数组长度
- push() 向数组最后一个元素后面添加一个元素
  - 向数组最后一个元素后面添加一个元素 返回数组长度

第六节 实战必备技能之数组常见操作讲解(二)

- **concat() 合并两个或多个数组**
  - **不会改变原有数组，返回一个合并完的数组**
- **join(separator) 数组转字符串 可传入一个参数用于分隔元素**
  - **separator可选。指定要使用的分隔符。如果省略该参数，则使用逗号作为分隔符。**
- **split(separator,howmany) 字符串转数组 可传入两个参数**
  - **separator必选，实现方法为将字符串按某个字符切割成若干个字符串，并以数组形式返回**
  - **howmany可选，该参数可指定返回的数组的最大长度。如果设置了该参数，返回的子串不会多于这个参数指定的数组。如果没有设置该参数，整个字符串都会被分割，不考虑它的长度。**
- **reverse() 翻转数组 颠倒数组中元素的顺序**

第七节 实战必备技能之数组常见操作讲解(三)

- **sort(sortfun) 数组排序 可传入一个参数 直接改变原数组**
  - **sortfun 可选。规定排序顺序。必须是函数。不传即按照字符编码顺序排序**
- **slice(start,end) 返回数组中被选定的元素,不包含下标为end的元素 可传入两个参数**
  - **start 必需。规定从何处开始选取。如果是负数，那么它规定从数组尾部开始算起的位置。也就是说，-1 指最后一个元素，-2**

指倒数第二个元素，以此类推。
    - **end**可选。规定从何处结束选取。该参数是数组片断结束处的数组下标。如果没有指定该参数，那么切分的数组包含从 **start** 到数组结束的所有元素。如果这个参数是负数，那么它规定的是从数组尾部开始算起的元素。
- **splice(index,howmany,x,y,z)**
    - 可以对数组进行增加，删除，更改的操作。非常灵活。会返回被删除的项目
    - **index**必需。整数，规定添加/删除项目的位置，使用负数可从数组结尾处规定位置。
    - **howmany**必需。要删除的项目数量。如果设置为 **0**，则不会删除项目。
    - **x,y,z**为可选项，为添加的新项目
- **indexOf(item,start)** 返回元素的索引，即下标，如果没有查找的元素就返回**-1**
    - **item**必须，查找的元素。
    - **start**可选，整数，规定在数组中开始检索的位置，如不选则为第一个开始检索

第六章 定时器讲解

第一节 讲解间隔性与延时性定时器的区别

- 间隔型定时器 setInterval(fun,time)
  - fun为执行的函数
  - time为间隔执行的时间，单位为毫秒，每过time时间就执行一次fun里面的代码
- 延时型定时器 setTimeout(fun,time)
  - fun为执行的函数
  - time为延时执行的时间，单位为毫秒，经过time时间后执行fun代码，只执行一次

第二节 讲解如何停止定时器

- clearInterval 清除间隔性定时器
- clearTimeout 清除延时性定时器

第三节 利用延时性定时器制作延时提示框

- 描述场景
  - 当我们鼠标移入小div的时候，会有一个弹窗大div显示在隔壁，鼠标离开两个div的时候弹窗div延时消失

第四节 利用间隔性定时器制作无缝滚动图片展示

- 具体代码请看课程源代码

愿景："让编程不在难学，让技术与生活更加有趣"

第七章 DOM操作讲解

第一节 讲解DOM节点的获取(一)及判断节点类型

- DOM获取子节点
  - childNodes
    - 使用childNodes获取子节点的时候，childNodes返回的是子节点的集合，是一个数组的格式。他会把换行和空格也当成是节点信息。
  - children
    - children不会返回文本节点，返回的是元素节点
- DOM获取父节点
  - parentNode
    - 获取的是当前元素的直接父元素，是w3c的标准
  - parentElement

        - 获取的是当前元素的直接父元素，是ie的标准
    - offsetParent
        - 获取离当前节点最近的一个有定位的父节点或祖先节点，如果没有即为获取body
- 判断节点的类型
    - nodeType 1为元素节点 3为文本节点(即空格或换行，字符)

第二节 讲解DOM节点的获取(二)

- 获取第一个子节点
    - firstChild 有可能会匹配到是空白或换行的节点信息
    - firstElementChild 不会匹配空白或换行的节点信息(获取到的是元素节点，不会获取到文本节点)
- 获取最后一个子节点
    - lastChild 有可能会匹配到是空白或换行的节点信息
    - lastElementChild 不会匹配空白或换行的节点信息(获取到的是元素节点，不会获取到文本节点)
- 获取兄弟节点
    - previousSibling和previousElementSibling 获取上一个兄弟节点
        - previousSibling 有可能会匹配到是空白或换行的节点信息
        - previousElementSibling 不会匹配空白或换行的节点信息(获取到的是元素节点，不会获取到文本节点)
    - nextSibling和nextElementSibling 获取下一个兄弟节点
        - nextSibling 有可能会匹配到是空白或换行的节点信息

        - nextElementSibling 不会匹配空白或换行的节点信息(获取到的是元素节点，不会获取到文本节点)

第三节 元素属性操作的三种方式

- 第一种：oDiv.style.display=“block”;
- 第二种：oDiv.style[“display”]=“block”;
- 第三种：Dom方式
    - 获取：getAttribute(属性名称)
    - 设置：setAttribute(属性名称, 值)
    - 删除：removeAttribute(属性名称)

第四节 详细讲解DOM节点的创建、插入和删除

- 创建并添加DOM元素
    - document.createElement('标签名') 创建一个DOM节点
    - appendChild(DOM节点) 为某个DOM添加一个子节点,在最后面添加
- 插入元素
    - insertBefore(节点, 原有节点) 在已有元素前插入
- 删除DOM元素

- **removeChild(节点) 删除一个节点**

第五节 讲解什么是文档碎片以及文档碎片的作用

- **document.createDocumentFragment() 创建文档碎片节点 是一个虚拟的节点对象**
- **文档碎片存在的意义就是减少页面渲染DOM的次数**
- **文档碎片可以提高DOM操作性能(理论上)**
  - **现在的硬件已经足够强大了，不像以前的硬件水平了**

```
<ul id="ul1"></ul>
//不使用文档碎片，每次插入li到ul都会造成页面重新渲染

var oUl=document.getElementById('ul1');

  for(var i=0;i<100;i++)
  {
    var oLi=document.createElement('li');

    oUl.appendChild(oLi);
  }

//使用文档碎片，每次把li先插入虚拟的节点文档碎片节点，最后只插入一次即可，只会重新渲染一次页面
  var oUl=document.getElementById('ul1');
  var oFrag=document.createDocumentFragment();

  for(var i=0;i<100;i++)
```

```
{
  var oLi=document.createElement('li');

  oFrag.appendChild(oLi);
}

oUl.appendChild(oFrag);
```

第六节 实战应用之DOM快速操作表格元素

- 首先获取到我们的表格，暂时用table1代表整个表格
- DOM来获取表头表尾、
  - 因为表头表尾只能存在一个 ，获取表头table1.tHead 获取表尾table1.tFoot
- DOM获取表体即
  - table1.tBodies获取到的是一个集合，因为可以存在多个tbody
- 获取行数的集合，即需要获取表格对象里的tr行数的集合
  - rows
- 获取行里面单元格td的数量 是一个集合 所以获取内部第一个需要cells[0]获取
  - cells

第八章 javascript的运动框架

第一节 深入了解JS的运动基础

- 先让div动起来，动起来之后设置一个条件让他停止运动，要不然他会一直运动下去

第二节 深度讲解单物体运动框架

- 封装好一个方法，让外部传入目的地的值，即让物体停下来的值

第三节 深度讲解多物体运动框架

- **封装一个方法让多个物体共用,本节讲解变速运动**
- **注意变速运动要做向上向下取整的操作**
- **注意再重新启动定时器是要把之前的定时器清除**
- **多物体运动每个物体要有自己的定时器，共用会出现bug**

第四节 深度讲解改变单一任意值运动框架

- 在多物体运动框架的前提下增加一个修改样式的参数

第五节 深度讲解链式运动框架

- 在改变单一任意值运动框架添加一个回调函数参数
- 链式运动顾名思义就是先运动完一个值再运动另一个

第六节 深度讲解改变多值运动框架

- 改变多值的运动框架意思是同时改变多个我们需要改变的样式值，不像链式操作那样需要等待前一个完成才可以执行下一个

第七节 完美运动框架展示

- 在多值运动框架上面考虑运动完之后传入回调函数
- 注意改变透明度opacity需要做特殊处理

愿景："让编程不在难学，让技术与生活更加有趣"

第九章 深入了解javascript事件

第一节 深度讲解事件对象event和事件冒泡

- 什么是事件对象
  - 用来获取事件的详细信息，例如：鼠标位置、键盘按键
  - ie下可以直接使用event，非ie需要传入事件对象参数
- 事件冒泡
  - 事件流会从DOM节点最里面的事件往最外面执行
  - 取消事件冒泡兼容写法
    - 获取事件对象 var oEvent=ev||event;
    - 取消事件冒泡 oEvent.cancelBubble=true

第二节 事件绑定、事件解绑及事件捕获讲解

- 事件绑定 addEventListener(事件名称,函数, 捕获) 第三个参数为可选，默认为false，false即事件流为事件冒泡
- 事件解绑 removeEventListener(事件名称, 函数, 捕获) ，默认为false，false即事件流为事件冒泡
- 事件捕获 和事件冒泡相反，如果开启了事件捕获就先执行捕获的事件再执行当前被点击事件
- 注意，事件绑定中传入匿名函数会导致无法进行事件解绑

第三节 实战必备之了解键盘事件及键盘码

- keydown 监听键盘按键按下事件
- keyup 用来监听键盘按键抬起事件，一般在使用组合键的时候可以用到
- 键盘码 keyCode

第四节 讲解默认行为及阻止默认行为

- 在浏览器里鼠标右击事件oncontextmenu会默认的给我们显示一个弹窗，这些就是事件的默认行为
- a标签跳转也是一个默认行为，我们可以通过在他的点击事件里面用return false来阻止默认行为

愿景："让编程不在难学，让技术与生活更加有趣"

第十章 Ajax使用

第一节 讲解何为Ajax及如何使用Ajax

- AJAX是什么?
  - AJAX 是一种在无需重新加载整个网页的情况下，能够更新部分网页的技术
  - AJAX 是一种用于创建快速动态网页的技术。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。
- 使用AJAX

- 调用封装好的AJAX，传入读取的文件地址，成功需要做的事情和失败需要做的事情
  - ajax(url,fnSucceed,fnFailed)
- npm install -g http-server 安装http-server 启动一个本地静态服务
- 浏览器为了安全性考虑，默认禁止跨域访问

第二节 深度剖析如何创建及使用Ajax

- 创建AJAX对象
  - Ie5、ie6使用var oAjax=new ActiveXObject("Microsoft.XMLHTTP")创建
  - 现代浏览器使用var oAjax=new XMLHttpRequest()创建(一般使用这种即可)
- 连接服务器
  - oAjax.open('GET', url, true);三个参数分别是请求的类型、文件在服务器上的位置、异步(true)或者同步(false)
- 发送请求
  - oAjax.send();
- 接收数据
  - onreadystatechange事件
    - readyState属性：请求状态

      0 （未初始化）还没有调用open()方法

      1 （载入）已调用send()方法，正在发送请求

      2 （载入完成）send()方法完成，已收到全部响应内容

3 （解析）正在解析响应内容

4 （完成）响应内容解析完成，可以在客户端调用了

- status属性：请求结果
  - 200: "OK"

    404: 未找到页面
- 请求成功返回内容responseText

愿景："让编程不在难学，让技术与生活更加有趣"

第十一章 初步了解面向对象

第一节 介绍js的面向对象的三大特性

- 什么是面向对象
  - 使用对象时，只关注对象提供的功能，不关注其内部细节
  - 面向对象是一种通用思想，不是只有编程中才能用，任何事情都可以用
- 封装性
  - 为了对代码更好的进行复用和维护
- 继承性
  - 解决了代码复用问题，为了代码更简单的复用
- 多态性
  - 多态的目的就是为了代码更灵活的复用，使子类继承的行为可

以变化

第二节 编写第一个面向对象程序之工厂函数(一)

- 对象的组成
  - 方法 ===> 函数：过程、动态的
  - 属性 ===> 变量：状态、静态的

第三节 编写第一个面向对象程序之工厂函数(二)

- 构造函数等同于我们此时只是做了初始化
- 我们还需要用new 来实例化我们的构造函数
  - new帮我们做了两件事
    - 帮我们创建空白对象 var this=new Object()
    - 帮我们返回经过加工过的对象 return this

第四节 深度剖析构造函数与原型配合实现面向对象编程

- 原型prototype
- 共用方法(行为)都写在原型上

第十二章 BOM介绍

第一节 BOM基础介绍

- 什么是BOM
  - BOM(Browser Object Model)即浏览器对象模型，是一个与浏览器窗口进行交互的对象
- 打开或关闭一个窗口
  - window.open()
  - window.close()

第二节 实战必备BOM常用属性和系统对话框

- window.navigator.userAgent 获取设备及浏览器的类型型号
- window.location 获取当前文档地址对象
  - location.host 主机加端口号
  - location.hostname 主机
  - location.port 端口号
  - location.protocol 协议
  - location.pathname 路径
  - location.hash 片段标识符，可以用于保存网页状态
  - location.search 返回问号后的字段

- 系统对话框
  - 警告框：alert(“内容”)
  - 选择框：confirm(“提问的内容”)
  - 输入框：prompt()
- 补充：window.close()可以直接关闭window.open()打开的窗口，如果关闭的是其他的窗口，浏览器出于安全性考虑会给你提示不能关闭

第三节 实战必备之BOM常用事件介绍

- window.onload
- window.onscroll
  - 滚动距离
    - 谷歌浏览器 document.body.scrollTop
    - IE、火狐浏览器 document.documentElement.scrollTop
- window.onresize
  - 可视区尺寸
    - document.documentElement.clientWidth
    - document.documentElement.clientHeight

第十三章 javascript回顾

第一节 回顾过去，展望未来

- 学习前端基础路线
  - **html+css ==> html5+css3 ==> js ==> vue第一季基础教学 ==> webpack ==> vue单页面电商实战项目教程 ==> 小程序**