学习方法

1. 键盘指法
2. 编程习惯
3. 大胆动手
4. 保持好奇心
5. 乐观面对错误

就业能力

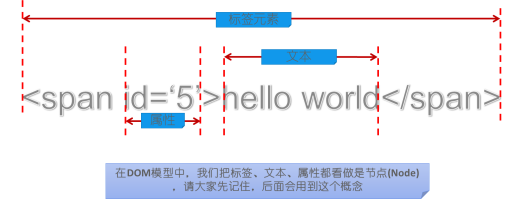
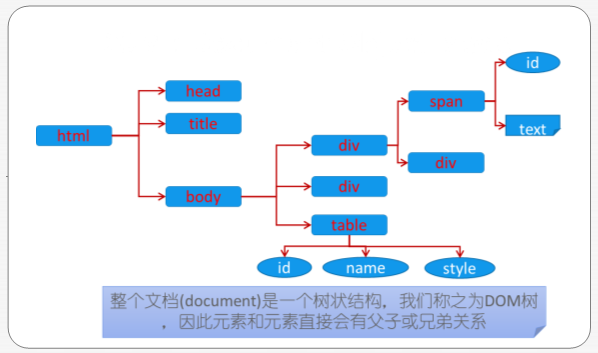
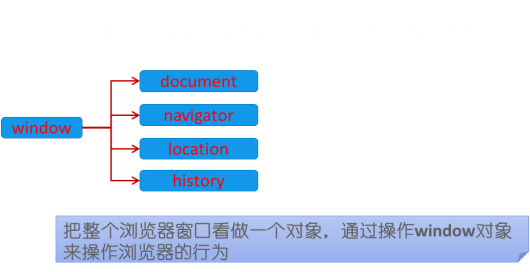
1. 分析能力
2. 独立思考能力
3. 程序阅读能力
4. 程序排错能力
5. 独立编写程序能力
6. 团队协作能力
7. 学习能力
8. 善于总结的能力

专业技能

JS基本语法

1. Javascript基本介绍及发展趋势
   1. JavaScript概述
      1. 概念：JavaScript是一种基于对象和事件驱动的客户端脚本语言
         1. JavaScript最初的设计是为了检验HTML表单输入的正确性。
         2. JavaScript起源于Netscape公司的LiveScript语言
      2. 作用：JavaScript 诞生于 1995 年。 它当时的目的是为了验证表单输入的验证。 因为在 JavaScript问世之前，表单的验证都是通过服务器端验证的。而当时都是电话拨号上网的年代，服务器验证数据是一件非常痛苦的事情。

1995 年， 当时工作在 Netscape(网景)公司的布兰登(Brendan Eich)为解决类似于 “向服务器提交数据之前验证”的问题。在 Netscape Navigator 2.0 与 Sun 公司联手开发一个称之为LiveScript 的脚本语言。为了营销便利，之后更名为JavaScript

* + 1. 标准：第一个标准的诞生ECMA-262（Jscript和JavaScript）
       1. 组成部分：
          1. 语法
          2. 类型
          3. 语句
          4. 关键字
          5. 保留字
          6. 操作符
          7. 对象
    2. JavaScript组成
       1. ECMAScript（核心）
       2. DOM：Document Object Model（文档对象模型）
       3. BOM：Browser Object Model（浏览器对象模型
  1. 发展趋势：  
     自从上世纪90年代中期发行以来，JavaScript就已经成为最流行的Web开发语言之一了。在2012年9月份，行业分析公司RedMonk就公布了JavaScript在开发语言里已经处于遥遥领先的地位。这主要归功于JavaScript提供丰富的、动态的网页内容的能力，以及JavaScript的轻便性和高易用性。

尽管JavaScript最初只是一个浏览器脚本语言，但是近几年，我们已经能够明显看到它在不断地演变：在桌面端、移动设备和服务端Web App领域都有所发展。在不久的将来，JavaScript有望成为IT企业里占据主导地位的开发语言——最终取代现在普遍使用的C，C++和Java开发语言。

1. JS和H5的关系  
    HTML5实际上并没有多少与HTML有关，它其实就是JavaScript。HTML本身有什么变化？不过一些新标签而已， 况且那个新标签都不难理解。HTML5的威力在于让你能用JavaScript来创建这些标签。假如没有后台代码通过Canvas来创建动画、游戏，或者 通过它来实现一些数据的可视化，这个标签也没有大用处。从浏览器开始支持Canvas开始，我已经看到了Asteroids（行星游戏）的上百个实现，那都是开发人员为熟悉这个新特性所做的练习。有的比较粗糙一些，而有的则极其精美。这些完全都要归功于JavaScript。  
    由此可见，HTML5并不是以尖括号为特征的标签语言的一次大的改进，其实质是赋予了JavaScript更强大的能力。WebGL库 （当前还羽翼未丰）支持在HTML5的画布中绘制实时的3D图形。HTML5的地理位置支持在浏览器中实现LBS（Location Based Service）应用——这都是手机的基本配置。而持久存储以及离线功能则为开发能与桌面应用媲美，但却在浏览器中运行的全功能应用奠定了基础。目前，就 连增加多点触摸事件的实验性的库也已经出现了。凡此种种，无一不是实实在在的JavaScript特性。HTML5只是为这些高级功能的发挥提供了舞台。  
    退一步讲，不依赖于HTML5的浏览器端开发库也取得了长足的进步。长久以来，JavaScript一直都是在HTML中实现动态效果的不二之选。可两个问题迟迟得不到解决：一是浏览器兼容性问题，二是直接操作DOM太麻烦。jQuery让这两个问题霎那间消失得无影无踪，这个库已经成为现代基于浏览器的客户端开发的基本配置。不过，并非只有jQuery。Protovis、还有D3，都可以让你直接在浏览器中创建复杂的交互性数据可视化效果，有史以来第一次让浏览器成为了展示数据的一个重要媒介。
2. 编写JS及如何运行JS
   1. 使用<script>…</script>
      1. <script>标签的属性
         1. Language=”javascript”已废弃。原来用于代码使用的脚本由于大多数浏览器忽略它，所以不用了。
         2. Src表示要引入的外部文件
         3. Type表示脚本语言的类型
      2. 输出方法
         1. document.write方法
            1. 在页面中输出内容
         2. Alert方法
            1. alert():带有一个确定按钮的警告框

如果括号中是数字就不须要加引号

如果是其它非数字数据要加引号（不区分单引号或双引号），在JS中只要加上引号就叫做字符串。引号嵌套时要注意不能交叉使用引号 ' "" ' " '' "JS语言中，所有的标点符号都是英文半角状态

* + - 1. console.log方法
         1. 在控制台输出，是程序员最常用的排错工具
      2. 如何输出换行
         1. 如果是document.write就用<br/>标签换行
         2. 如果是alert就用转义字符\n(表示换行)  
            \t :表示横向跳格（一次8个空格）
      3. 原样输出标签（&lt;代表< &gt;代表>）
    1. 注释
       1. 作用
          1. 给程序加解释文字
          2. 注销代码
       2. 表示符
          1. 单行注释 //
          2. 多行注释 /\* \*/
    2. 引入JS文件的方式
       1. 内部添加<script type = "text/javascript"></script>
       2. 外部引入js文件
       3. 注：
          1. 必须是双标签
          2. 从外部引入JS文件时，script标签中不允许写入任何内容，系统不会执行标签中的任何内容

<script type="text/javascript" src="js文件路径"></script>

1. 变量命名规则和关键字的介绍
   1. 标识（志）符：标识变量、数组、函数等的有效字符序列。
   2. 标识符的命名规则
      1. 名字中只能包含 字母、数字、下划线、$、（中文）
      2. 不能以数字开头
      3. 不能是保留字或关键字
      4. 具有相关含义
      5. 驼峰命名
         1. 大驼峰：如ZhongHuaRenMinGongHeGuo
         2. 小驼峰：如zhongHuaRenMinGongHeGuo
      6. 匈牙利命名法：
         1. Int iNum:以i开头的变量存储整数
         2. Float fNum:以f开头的变量存储小数
         3. Boolean bBoo:以b或bo开头的变量存储布尔变量
         4. String sStr:以s开头的变量存储一个字符串
         5. Char cCh:以c开头的变量存储一个字符
         6. Object obj:以o开头的变量存储对象
         7. Array arr:以arr开头的变量存储数组
         8. Function fn:以fn开头的变量存储函数
   3. 关键字和保留字：系统将一些特定字符用于特殊使用的，称为保留字。  
      关键字：  
      break else new var case finally return void catch for switch while   
      continue function this with default if throw delete in try do instanceof typeof  
      保留字：  
      abstract enum int short boolean export interface static byte extends long super   
      char final native synchronized class float package throws const goto private transient debugger implements protected volatile double import public
2. 变量的概念及基本运算
   1. 变量的概念：在内存中开辟一块空间，用于存储数据，且随着程序的运行，数据有可能发生变化，所以称为变量
   2. 变量的声明：
      1. 显示声明：var 变量名;（varible:变量）
         1. var 变量名 = 常量/变量/表达式;
            1. 如：var num;  
                var num = 3 + 2;
         2. 显示声明可一次声明多个变量，变量之间以逗号（英文半角）分隔
            1. 如：var numOne,str = “”,bo = false;
      2. 隐式声明：变量名 = 常量/变量/表达式;
         1. 如：num = 4 / 2;
      3. 变量的赋值：
         1. 变量的初始化：定义变量的同时就给变量赋值
         2. 先声明后赋值
      4. 松散类型（弱类型）：声明变量不须要确定是什么类型的变量。

强类型：C语言 int num;

* 1. 算术运算符
     1. 表达式：由变量、常量和操作符组成。单独一个变量或常量也叫表达式
     2. 算术运算符
        1. \* ：乘
           1. 数字类型相乘，正常相乘
           2. 数字(字符串）类型乘以字符串（数字）类型，先将字符串自动转换为数字，如果可以转换，则正常相乘，如果转换不了数字，结果是NaN
           3. 乘以布尔值时，先将true转为1 false转为0 再计算
           4. 乘以null时，先将nulll转为0再计算
           5. 乘以undefined时，先将undefined转为数字，不能转，所以转成了NaN，结果就是NaN
           6. 如//console.log(4\*'4'); //16

//console.log(4\*'a'); //NaN:Not a Number 类型是：Number

//console.log('4'\*'4a');

//console.log(4 \* true); //4

//console.log(4 \* false); //0

//console.log(4 \* null); //0

//console.log(4 \* undefined); //NaN

* + - 1. / :除
         1. 规则同\*
         2. 注：console.log(0/4);  
            console.log(4/0); //Infinity无穷  
            console.log(0/0); //NaN
      2. % ：模（取余）
         1. 规则同\*
         2. 如：console.log(4%3); //1  
            console.log(4%2); //0  
            console.log(4%5); //4  
            console.log(4%6); //4  
            console.log(4%3.3);
      3. - ：减
         1. 规则同\*
         2. 如：console.log(8 - '4');  
            console.log(8 - '4a');  
            console.log(8 - true);  
            console.log(8 - false);  
            console.log(8 - null);  
            console.log(8 - undefined);
      4. + ：加
         1. 如果加号两边有一个是字符串，就进行连接作用
         2. 数字（不包含NaN） 加上 true (1) 、false(0) 、null(0)
         3. 数字（包含NaN）加上undefined、NaN，结果都是NaN
         4. 如: document.write(3 + "3");  
            document.write("<br/>");  
            document.write(3 + true + false + null);  
            document.write("<br/>");  
            document.write(3 + NaN);  
            document.write("<br/>");  
            document.write(3 + undefined);  
            //document.write(3 + true + "false");  
            document.write("4" + 4 + 8 + null + undefined);

第二节：应用

1. 为抵抗洪水，战士连续作战89小时，编程计算共多少天零多少小时？
2. 小明要到美国旅游，可是那里的温度是以华氏度为单位记录的。  
   它需要一个程序将华氏温度（80度）转换为摄氏度，并以华氏度和摄氏度为单位分别显示该温度。  
   提示：摄氏度与芈氏度的转换公式为：摄氏度 = 5/9.0\*(华氏度-32)保留3位小数

第三节

1. 扩展：数据类型
   1. 基本数据类型
      1. Number：数字类型 3 -3.3 64位浮点数形式存储（相当于C语言中的Double）
      2. String:字符串类型 "ab" 'bc' 只要加引号的就是字符串
      3. Boolean:布尔类型 true(真) false(假)
   2. 特殊数据类型
      1. null:表示空，它的类型是Object（对象）
      2. undefined:未定义 它的类型是Undefined类型
   3. 复杂数据类型
      1. Array：数组 它的类型是Object(对象）
      2. Object：对象
   4. 官方类型标准：
      1. 简单类型：Number/String/Boolean/null/undefined
      2. 复杂类型：Object
2. 自增自减运算符
   1. ++ ：递增（每次加1） //常用在循环语句中。
   2. - - :递减（每次减1）
   3. 运算逻辑：
      1. i++:（后缀自增）先取出变量i中的值参与运算，再将i的值加1
      2. ++i:(前缀自增）先将i的值加1，再取i的值参与运算
      3. i--与--i同上
3. 关系运算符（结果是布尔值：true、false)
   1. > :大于
      1. 数字类型比较：正常比较
      2. 数字类型和字符串类型比较：将字符串转成数字，如果可以正常转成数字，正常比较，如果不能转成数字，则转成NaN，结果为false
      3. 字符串类型比较，依次比较
      4. 一律不能正常比较的，结果都是false
   2. < ：小于
      1. 规则同大于
   3. >= :大于或等于
      1. 规则同大于
   4. <= :小于或等于
      1. 规则同大于
   5. == ：等于
   6. != :不等于 //只比较转换后的结果
   7. === ： 恒等于（全等于）
   8. !== :不恒等（不全等） //先比较类型，再比较结果
   9. 注：NaN不等于任何数，包含它自己
   10. 如：alert(4 > 3);  
       alert(4 > '3');  
       alert(4 > "100");  
       alert(4 > 'a');  
       alert(98 > 'a');  
       alert("2" > "100");  
       alert("我">"你");  
       alert(4 > true)  
       alert(4 > undefined);  
       alert(4 > NaN);  
       alert( 3 >= 3);  
       alert(3 == '3');  
       alert(3 === '3');  
       alert(false == null);  
       alert(NaN == NaN);  
       alert(NaN === NaN);
   11. 切记：
       1. alert(null == undefined); //true
       2. alert(null == ""); //false
       3. alert(null == 0); //false
       4. alert(null == false); //false
       5. alert(null === undefined); // false
4. 逻辑运算符：0:表示假，其它非0的数据表示真
   1. ！ ：非真即假，非假即真
      1. 如：alert(!0);   
         alert(!1);   
         alert(!-2);   
         alert(!null);   
         alert(!undefined);   
         alert(!NaN);   
         alert(!"中");   
         alert(!(3>='3'));   
         alert(!3 === '0');   
         alert(!0 !==1);   
         alert(!(null == undefined));   
         alert(!(null == ""));  
         alert(!(null == 0));
   2. &&:与（并且）
      1. 逻辑关系：先判断&&左边的表达式，如果表达式的值为false,则返回左边表达式的值（此时为&&的短路问题）。如果表达式的值为true，则继续判断右边表达式的值，返回右边表达式的值。
      2. 如：alert(3-3&&8);   
         alert(3===3&&8);   
         var a = 3;   
         alert(!(3===3)&&(a = 8));   
         console.log(a);   
         var a = 3;  
         alert(!(3==='3')&&(a = 8));   
         console.log(a);
   3. ||:或（或者）
      1. 逻辑关系：先判断||左边的表达式，如果表达式值为true时，则返回左边表达式的值（此时，为|| 的短路问题)。如果表达式值为false时，继续判断右边表达式的值，返回右边表达式的值。
      2. 如：var a = 3;  
         alert(!(3==='3')||(a = 8));   
         console.log(a);   
         var a = 3;  
         console.log(!(3===3)||(a = 8));   
         console.log(a);
   4. 练习：
      1. 输出下列表达式的值。
         1. (100>3)&&(‘a’>’c’)
         2. (100>3)||(‘a’>’c’)
         3. !(100<3)
      2. 构造一个表达式来表示下列条件：
         1. number等于或大于90，但小于100
         2. ch不是字符q也不是字符k
         3. number界于1到9之间（包括1不包括9），但是不等于5
         4. number不在1到9之间
      3. 判断这个字符是空格，是数字，是字母
      4. 有3个整数a,b,c，判断谁最大，列出所有可能
      5. 判断year表示的某一年是否为闰年，用逻辑表达式表示。  
         闰年的条件是符合下面二者之一：
         1. 能被4整除但不能被100整除
         2. 能被400整除
5. 三元运算符（三目运算符、条件运算符）
   1. 格式：（条件）？表达式1：表达式2
      1. 如果（条件成立）？则执行表达式1；否则：执行表达式2
   2. 逻辑关系：先计算条件的值，值为true时，执行？后面的表达式1，值为false时，执行：后面的表达式2
   3. Demo:
      1. 求一个三位数的和
      2. 判断一个年份是闰年还是平年
         1. 能被4整除但不能被100整除或能被400整除的年份是闰年
      3. 判断一个数是偶数还是奇数
      4. 判断一个数是正数还是负数
6. 赋值运算符
   1. 简单赋值：= 将赋值号右边表达式的值存入左边变量
   2. 复合算术赋值：+= -= \*= /= %=
   3. 运算逻辑：取运算符左边变量中的值与运算符右边表达式的值进行相应的算术运算，运算后的结果再次赋值给左边的变量。
      1. 如：  
         var a = 5;  
         a += 4; //a = a + 4;
   4. 练习：  
      var i=5;  
      var a=3;  
      a+=++i + ++i - i++ + i++;  
      var b=a++ + i++ + ++a + a++ + "a++";  
      求：i、a、b的值分别是多少？
7. 特殊运算符
   1. typeof ： 检测简单数据类型
      1. 如：console.log(typeof -4); //Number 返回值”number"  
         console.log(typeof ""); //String 返回值"string"  
         console.log(typeof false); //Boolean 返回值"boolean"  
         console.log(typeof null); //Object 返回值 "object"  
         console.log(typeof undefined); //Undefined 返回值：'undefined'  
         console.log(typeof typeof 3); //String "string"  
         //typeof 3 "number"  
         //typeof "number" string
   2. new :创建对象
      1. 如：var oNum = new Number();(数字对象)  
         var oStr = new String(); (字符串对象)  
         var oBoo = new Boolean(); (布尔对象)  
         var oArr = new Array(); (数组对象)  
         var oRe = new RegEXP(); (正则表达式对象)  
         var obj = new Object(); (对象)
8. 类型的转换
   1. 自动类型转换
   2. 强制类型转换
      1. isNaN();判断括号中的表达式的值是否是NaN,如果是NaN,则返回true,否则返回false
         1. 如：//console.log(isNaN("love"));  
            //console.log(isNaN("888 666"));  
            console.log(isNaN("3"+undefined));
      2. parseInt():将字符串强制转换成整数（必须以有效数字开头才会转成数字，否则转成NaN）
         1. 如：console.log(parseInt("23.45abc"));  
            console.log(parseInt("abc123"));
      3. parseFloat():将字符串强制转换成小数（必须以有效数字开头才会转成小数，否则转成NaN）
         1. 如：console.log(parseFloat("34.45.56.abc"));  
            console.log(parseFloat("sa33"));
      4. Number():将字符串强制转换成数字（整个字符串必须是有效的数字才可以转，否则转成NaN）
         1. 如：console.log(Number("34.23"));  
            console.log(Number("23 34"));  
            console.log(Number("2.3.4"));
      5. 扩展：eval():将字符串强制转换成表达式并返回表达式的值
         1. console.log(eval("3+4"));  
            console.log(eval("3+undefined"));
9. 八进制和十六进制的介绍
   1. 八进制：逢八进一
   2. 十六进制：逢十六进一
      1. 0 1 2 3 4 5 6 7 8 9 A B C D E F
10. 扩展：代码规范
    1. 代码缩进（JS/html）
    2. 运算符两边的空格
    3. 语句最后的;号
    4. 小括号的嵌套要加空格
    5. 双引号、单引号嵌套 “ ‘’ ” ‘ “” ‘

第四节：应用

1. 计算两个文本框的和
2. var k=0; console.log(k++ + ++k +k +k++)
3. 掌握逻辑运算的意义
4. 掌握八进制、十六进制的写法
5. 掌握NaN的含义
6. 掌握Number函数的转换结果

第五节：综合应用

1. 计算两个文本框的加减乘除  
    要求：1）使用parseInt方法类型转换  
    2）计算结果使用Math.round方法四舍五入