**JS基础语法**

**一、JS概述**

**1. JS是什么**

* Javascript的简写，是客户端的脚本语言
* 组成
  + ECMAScript，描述了该语言的语法和基本对象
  + 文档对象模型（DOM），描述处理网页内容的方法和接口。
  + 浏览器对象模型（BOM），描述与浏览器进行交互的方法和接口。
* js的作用
  + 前台的动态效果
  + 页面局部刷新
  + 前后台数据交换
* js嵌入方式

<html lang="en">  
<head>  
<meta charset="UTF-8">  
<title>Document</title>  
<script type='text/javascript' src='13.js'></script>  
<script>  
// alert("hello world")   
</script>  
​  
</head>  
<body>  
  <div></div>  
<button onclick='alert("world")'>click</button>  
</body>  
</html>  
<script>  
// alert("你好")   
</script>

* + 内部
  + 外部
  + 行内
* js操作三要素：
  + 获取元素
  + 添加事件
  + 再操作

**二、JS数据类型和变量**

js是一种弱类型的语言

**1. 变量声明**

* 声明方式
  + 隐式声明
  + 显式声明
  + 注意：**变量可以重复定义**

//变量的显示声明  
var iNum = 10;  
var a = b = c = 20;  
​  
//隐式声明  
d = 20;

* 命名规则
* 数字、字母、下划线、$组成，不能以数字开头
* 区分大小写
* 不能使用保留字
* 采用匈牙利命名法，第一个字母表示类型

int   bool     string   obj     array  
iAge   bFlag   sName     oDiv   aDivs

* 输出

//变量的显示声明var iNum = 10;var a = b = c = 20;​ //隐式声明 d = 20;

//1 在控制台输出，请使用谷歌或火狐浏览器查看  
console.log(iNum,a,b,c,d);  
​  
//2 弹窗输出  
alert(d);  
​  
//3 当前文档流输出，显示到页面上  
document.write(a,' ',b,' ',c);

**2. 注释方式**

* 行注释 //
* 块注释 /\*\*/ 不能够嵌套

**3.数据类型**

js包含了以下数据类型：数值（Number）、布尔（boolean）、字符串（string）、数组(array)、对象(object)、未定义(undefined)、null、NaN

* 可以使用运算符typeof来查看变量或常量的类型
* 数值 整数和浮点数都是数值类型
* 布尔 true和false
* 字符串
  + 单引号和双引号引起来的文本都是字符串
  + 单引号可以套双引号，双引号可以套单引号
  + 字符串拼接使用“+”运算符，拼接时其他类型都转为字符串
* 未定义(undefined)

var a1;  
console.log(a1); //undefined  
​  
//函数定义未返回值  
function test()  
{  
console.log('hello');  
}  
a = test();  
console.log(a);  
​  
//对象没有的属性  
var obj = new Object;  
console.log(obj.name);

* NaN(not a number)
  + 运算错误的时候会产生NaN
  + 不带表任何数值，和任何数都不等，包括自己
  + 判断是否是NaN使用isNaN()

**三、运算**

**注意：**

* js中所有的变量都可以看成对象

**类型转换**

1. 转换为false
   * undefined
   * 0和-0,0.0
   * ''
   * null
   * NaN
2. 转换为字符串
   * 任何类型和字符相加都会转换为字符串
   * 布尔、数值类型都有toString([radix])方法,radix可以指定转换的进制
   * 也可以是用字符串构造方法String(x),将x转换为字符串
3. 转换为数值
   * parseInt可以将字符串转为整数
   * parsefloat可以将字符串转为浮点数
   * Number（字符串）转换为Number
   * 字符串转数值时，只转开头部分的数字，如果开头没数字，产生NaN
   * 字符串转为数值时，开头的+、-和空格可以正确处理
   * Number转换是字符串中不能包含其他字符否则NaN
4. 逻辑运算
   * 逻辑与运算：a && b，当a为真，返回b的值（不转换为boolean）,否则如果a为假，结果a的值
   * 逻辑或运算 a || b，相当于a?a:b,a为真，结果为a，否则结果为b
   * 注意短路计算

**运算符**

1. 算数运算（+、-、\*、/、%）
2. 自增自减
3. 赋值运算
4. 关系运算
   * ===和!== 判恒等
   * ==和!=
5. 逻辑运算
   * || 逻辑或
   * && 逻辑与
   * ! 逻辑非
6. 三元运算(?:)

b = 第一个表达式? 第二个表达式:第三个表达式  
首先计算第一个表达式的值，如果为真，将第二个表达式得值赋给b，否则将第三个表达式的值赋给b

1. 逗号

表达式1,表达式2,表达式3.....表达式n  
结果是表达式n的值

**注意：**

1. 逗号运算符，优先级最低，从左向右算，整个表达式的值是最后一个
2. 其他运算，需要使用函数，比如数学函数：random()产生0~1随机值
3. 短路计算，对于逻辑或和逻辑与来说，如果逻辑与第一个表达式为假，第二个表达式不计算，对于逻辑或来说如果第一个表达式为真，第二个表达式不计算

**四、流程控制**

1. 分支
   * 单if
   * if-else
   * if -else if
   * switch-case
2. 循环
   * while
   * for
   * do while
   * for-in 遍历对象/数组

for ( 变量 in 对象/数组 ){  
   <语句组>  
}

1. break和continue
2. 异常处理

try {  
   //可能异常语句  
   //throw 异常  
} catch (e) {  
   //异常处理  
   alert(e.message);//显示错误信息  
} finally {  
   //无论如何总会执行的代码,finally是可选的  
}

* 注意：
  + 可以捕获系统异常和用户抛出的异常
  + 只能有一个catch
  + throw 后面的异常可以为字符串、整数、逻辑值或者对象

1. with语句 打开对象

with (<对象>){  
　　<语句组>//可以直接使用对象的方法  
｝  
​  
with (document) {  
   write ("限时抢购物品：");  
   write ("<Li>ViewSonic 17\" 显示器。");  
   write ("<Li>EPSON 打印机。");  
}

**五、数组**

**1.数组定义方式：**

var a = Array();  
var b = [];  
var c = Array(1,2,3);  
var d = [1,211,-3,4,'5',true];//元素不必是同一个类型

* 注意：
  + 数组长度可以用length获取：a.length
  + 数组下标从0开始
  + 下标越界不会报错，只是元素的值是undefined
  + 数组大小可变

**2.数组的访问**

* 通过下标访问数组元素
* 数组遍历
* 不会产生越界的问题，如果访问的数组元素不存在，返回undefined

**3.多维数组**

a = [[1,2,3],[3,4,5],[5,6,7]];  
console.log(a[0][1]);

**4.数组常用函数**

a=[1,22,3,44,8]  
a.push(10);//末尾添加  
a.shift(9,10);//开头添加  
console.log(a.join(',));//'1,22,3,44,8'

**注意**

* 点和[]的区别：.后面不能跟变量，[]可以是变量；能使用.地方都可以使用[],反之则不然

**六、函数**

**1.函数定义的两种形式：**

//1 命名函数  
function test(a,b)   
{  
return a + b;  
}  
//2 匿名函数  
function(a,b)   
{  
  return a + b;  
}

* 注意：
  + 函数名区分大小写
  + 函数可以重写，后面的覆盖前面的

**2.函数调用**

console.log(test(3,5))  
function test(a,b) {  
return a + b;  
}

* 直接调用
* 调用可以在定义之前
* 回调函数时，可以传递函数名（不是字符串）,匿名函数、变量函数

**3.封闭空间**

(function(a){  
  console.log(a)  
})(10);

**4. 函数参数**

* 函数参数不用加var，不要加类型
* 如果函数有形参，可以不传实参，或传多个实参
* 可以通过arguments获得函数实参,arguments不是数组，虽然可以遍历，可以有length​ //将arguments转换为数组​ var arr = Array.prototype.slice.call(arguments);

**5. 参数传递**

* 基本类型是值传递
* 复合类型（对象，数组、函数）也是值传递，是引用传递

**6. 内部函数**

* 内部函数定义在其他函数内部，外部无法调用
* 外部函数调用多次

**7.递归函数**

**8.变量的作用域(\*\*\*)**

* 在函数内部的用var定义变量，是局部变量，作用域在整个函数体内
* 在函数外部定义的变量叫全局变量
* 函数体内可以直接引用全局变量
* 如果局部变量和全局变量重名，局部优先
* 使用var定义的变量和不用var定义的变量的区别
  + 在函数内部，var定义的是局部变量，不用var是全局变量

**七、面向对象**

//1 定义对象  
obj1 = new Object()  
obj1.name = "大傻"  
obj1.play = function() {  
this.age = 20  
console.log("投降输一半")  
}  
obj1.play()  
​  
//2 工厂模式  
function createCat(name,age){  
cat = new Object  
cat.name = name  
cat.age = age  
cat.miao = function(){  
console.log("傻子，快把本秒的小鱼干拿过来")  
}  
return cat  
}  
obj2 = createCat('大脸猫',4)  
obj2.miao()  
​  
//3 构造函数  
function Person(name,age){  
this.name = name  
this.age = age  
this.walk = function(){  
console.log("走起")  
}  
}  
p1 = new Person("哈哈",20)  
p1.walk()  
​  
//4 原型法构造  
function Person(name,age){  
this.name = name  
this.age = age  
}  
Person.prototype.walk = function() {  
console.log("走")  
};  
p2 = new Person("胡汉三",30)  
p2.walk()  
​  
// 5 json对象  
obj3 = {  
name:'tom',  
age:21,  
eat:function(){console.log("吃")}  
}  
obj3.eat()  
console.log(obj3)

**作业**

1. 把数组函数、数学函数、字符串函数、日期函数验证一遍
2. 写一个例子，3个按钮，修改div的背景颜色、显示消失、字体颜色
3. 写一个简单的四则计算器，输入两个数，一个运算符，点击计算按钮，弹出计算结果