基本小常识

所有的 JS 代码都必须写在</script>…内容…</script>标签里

script 标签一般都是写在<head>中

规律：

<1>type = "text/javascript" 说明当前 script 标签中文本的类型

<2>为了语法规范，script 标签写在 head 标签中。

<3>可以引入多个 script 标签，多个 script 标签之间，顺序执行。

<4>js 代码可以外部引入 src 引入外部文件。

<5>如果当前 script 标签作用引入外部文件，这个 script 标签中， 就不能再写代码了。

生成 html 模版 快捷键

！+tab 键

注释 快捷键

单行注释：//…内容…（快捷键：ctrl+/）

多行注释：/*…内容…*/（快捷键：ctrl+shift+/）

基本数据类型

<1>数字 number 100 3.14

<2>字符串 string 所有带双引号/单引号 'hello' "hello"

<3>布尔值 boolean true false

<4>特殊数据类型 null 空 undef ined 未声明

标识符：

用户自定义的所有名字叫做标识符。变量名

规律:

1、标识符必须由数字、字母、下划线和美元符号($)组成。

2、不能以数字开头。

3、标识符区分大小写，age 和 Age 这是两个变量

4、标识符必须见名思意。

基础 代码

引入外部标签 代码

<script type = "text/ javascript" src = "demo. js">空</script>

//scr 为外部标签文件路径

声明变量(并且初始化)

```
var  变量="xxx",变量="xxx";
```

打印 代码(在网页中打印)

```
document.write("打印内容")
//打印中文需要加上引号""
```

警告框 代码

```
alert("…输出内容…");
```

输入当前变量/常量的数据类型：alert(typeof 变量名);

输入格式扩展： alert("输出内容"+变量+"输出内容"+变量);

语句 代码

单分支 if 语句格式：

```
if(条件)
{
    条件成立将执行的代码;
}
//注意事项：无
```

多分支语句 else if 语句格式：

```
if(条件 1)
{
    条件 1 成立将执行的代码;
}
else if(条件 2)
{
    条件 2 成立将执行的代码;
}
else
{
    都不符合上述条件，将执行这个代码;
}
//注意事项：语句按顺序一条一条执行
```

开关语句 switch 语句格式：

```
switch(变量)
    case"常量 1"：满足 常量 1 将执行语句；break；
    case"常量 2"：满足 常量 2 将执行语句；break；
            ……
    case"常量 n"：满足 常量 n 将执行语句；break；
    default：都不满足上述常量将要执行的语句；break；
    //注意事项：break 是一定不能省略的，如果省略将会造成事件穿透
```

循环 while 语句格式

```
while(循环条件)
{
条件成立将执行的代码；
}
//注意事项：1.循环条件中使用的变量需要经过初始化
2.循环体中应有结束循环的条件，否则会造成死循环
```

do while 循环 语句格式

```
do
{
循环体；
}
while(循环条件)
//注意事项：循环条件成立将继续执行循环体
```

for 循环 语句格式

```
for(变量初始化；循环条件；表达式)
{
循环语句；   //将在循环成立时执行
}
//注意事项：1.变量初始化：在 for 语句中只在刚进入循环时执行一次
2.表达式：将在循环执行完一次之后执行一次，在循环中起到计数作用 [i++]
```

break 语句格式

```
【使用格式】break；
//注意事项：直接跳出循环体，执行下一条语句
```

continue 语句格式

```
【使用格式】continue；
//注意事项：只能在循环语句中使用，使本次循环结束，即跳过循环体中下面尚未执
行的语句，接着进行下次是否执行循环的判断
```

重难点

事件的定义

onClick _ 单击事件
onMouseOver _ 鼠标经过事件
onMouseOut _ 鼠标移出事件
onChange _ 文本内容改变事件
onSelect _ 文本框选中事件
onFocus _ 光标聚集事件
onBlur _ 移开光标事件
onLoad _ 网页加载事件
onUnload _ 关闭网页事件

【格式】

<button type="button" onclick="调用自定义函数"> …内容… </button>

自定义函数

无参函数 定义格式

```
function 函数名( )
{
   函数体;
}
//注意事项：函数体内，外面怎么写 JS 代码，这里也如何去写 JS 代码
```

有参函数 定义格式

```
function 函数名(形式参数 1，形式参数 2)
{
函数体;
}
//注意事项：JS 是弱语言，可以不声明变量类型
return 【返回值】 使用方法
```

【注】可以通过 return 语句跟后面的要返回的值来实现返回值。

【注】return 后面所跟表达式的值，就是当前函数调用的值。

学习领悟：

自定义函数可以全都封装在一个**.js 文件中，然后再通过以下代码进行调用

<script type = "text/ javascript" src = " **. js">空</script>

计时器

循环执行

【格式】

```
<script type="text/javascript">
    var 变量名= setInterval(事件,间隔时间);      /* 开始循环格式 */
```

```
            clearInterval(变量名);                    /* 停止循环格式 */
        </script>
```

【释义】第一次事件的执行将在间隔数之后。

【示例】

```
<script type="text/javascript">
    var time = setInterval(function(){getTime();},1000);        /* 开始循环格式 */
    function getTime(){                                  /* 设置循环事件 */
        alert("我是循环内容！");
    }
    function StopTime(){                                 /* 停止循环方法 */
        clearInterval(time)
    }
</script>
```

延时执行

【格式】

```
<script type="text/javascript">
    var 变量名 = setTimeout(事件，延时时间);        /* 设置延时事件 */
    clearTimeout(变量名);              /* 停止延时事件 */
</script>
```

【示例】

```
<script type="text/javascript">
    Var demo = setTimeout(function(){alert("Hello！")},3000);  /* 设置延时事件 */
    clearTimeout(demo);              /* 停止延时事件 */
</script>
```

About 对象

创建自定义对象

【格式 1】

```
<script type="text/javascript">
    people = new Object();           /* 创建对象 */
      people.name = "Mr.温";
      people.age = "20";
      document.write("姓名：" + people.name + "年龄" + people.age)
      delete people.age;             /* 删除对象属性 */
</script>
```

日期对象

【创建格式】

```
var 对象名称 = new Date(参数);
```

【参数】不传入参数，默认获取当前系统时间

生成日期对象的参数类型可以是：

Date("2020/09/13");

Date("2020-09-13");

Date("2016,09,13,21,29");

【拓展】

```
var 变量名 = 对象名称.getFullYear();        /* 获取年份 */
var 变量名 = 对象名称. getHours ();         /* 获取小时 */
var 变量名 = 对象名称. getMinutes ();       /* 获取分钟 */
var 变量名 = 对象名称. getSeconds ();       /* 获取秒钟 */
var 变量名 = 对象名称. getTime ();          /* 获取毫秒 */
var 变量名 = 对象名称. getDay ();           /* 获取星期 */
```

Window 对象

获取浏览器窗口宽度

【格式】var X = window.innerWidth;

【释义】获取当前浏览器的宽度赋予变量 X

获取浏览器窗口高度

【格式】var Y = window.innerHeight;

【释义】获取当前浏览器的高度赋予变量 Y

打开新窗口

【格式】window.open("超链接/URL","新窗口名称","窗口属性");

【参数】

窗口属性：

```
Width = *px;          /* 设置新窗口宽度 */
Height = *px;         /* 设置新窗口高度 */
Top = 100;            /* 设置新窗口顶部位置 */
Left = 100;           /* 设置新窗口左部位置 */
                ..............................
```

【示例】

```
window.open("https://www.baidu.com/","Mr.温","  height=20px,width=10px,top=100,left=500");
```

关闭当前窗口

【格式】window.close();

History 对象

返回上一页

【格式】history.back();
【功能】类似于浏览器点击后退按钮

前往下一页

【格式】history. forward ();
【功能】类似于浏览器点击后退按钮

历史记录

【格式】history.go():
【功能】进入历史中的某个页面

Location 对象

获取主机域名

【格式】var x = window.location.hostname;

获取当前页面的路径和文件名称

【格式】var x = window.location. pathname;

获取 web 主机的端口

【格式】var x = window.location. port;

获取使用的 web 协议

【格式】var x = window.location. protocol;

重载新页面

【格式】location.assign("超链接/URL");
【功能】跳转到新的页面

Screen_对象

获取屏幕宽度
【格式】var x = screen.width;
获取屏幕高度
【格式】var x = screen.height;
获取屏幕可用宽度
【格式】var x = screen.availWidth;
获取屏幕可用高度

【格式】var x = screen.availHeight;

事件句柄

DOM 0 级事件

【事件句柄使用格式】

```
<script type="text/javascript">
    var 变量名 = document.getElementById("控件 ID");
    变量名.onclick = function(){          /* 设置 DOM 0 级事件 */
        JavaScript 语句;
    };
    变量名.onclick = null;                /* 清除事件 */
</script>
```

DOM 2 级事件

【事件句柄使用格式 _ 01】

```
<script type="text/javascript">
        /* 添加事件句柄 */
        document.getElementById("元素 ID").addEventListener("事件",自定义函数名);
        function 函数名(){
            Java 语句;
        }
        /* 移除事件句柄 */
        document.getElementById("元素 ID").removeEventListener("事件",自定义函数名);
</script>
```

【事件句柄使用格式 _ 02】

```
<script type="text/javascript">
        document.getElementById("元素 ID").addEventListener("事件",function(){
            JavaScript 语句;
        });
</script>
```

示例

```
<script type="text/javascript">
    document.getElementById("bth").addEventListener("click",function(){
        alert("事件句柄 _ 001！")
    });
</script>
```

数组的使用

数组的声明：

方法一： 使用 new 运算符创建数组

【格式】：var 数组名=new Array（数组元素 或 数组元素个数）；

```
//例：var abc=new Array(10);    //声明一个元素个数为 10 的数组 abc。
//例：var abc=new Array(1,true,"hello");  //声明一个含有元素 1，true，
                                          "heiio"元素的数组 abc
```

方法二：省略 new 运算符创建数组

【格式】：var 数组名= Array（数组元素 或 数组元素个数）；

```
//例：var abc = Array(10);  //声明一个元素个数为 10 的数组 abc。
//例：var abc = Array(1,true,"hello");      //声明一个含有元素 1，true，
                                            "heiio"元素的数组 abc
```

方法三：通过常量来创建数组【推荐】

【格式】：var 数组名 = [数组元素]；

```
//例：var abc = [ ];                        //声明一个空数组
//例：var abc=[1,true,"hello"];             //声明一个含有元素 1，true，
                                            "heiio"元素的数组 abc
```

数组元素的调用：

格式：数组名[数组的位置]；

```
//例：abc[n];     //调用了数组 abc 中的第 n+1 个元素
```

【注】数组长度访问：

【格式】：数组名.length　　//访问数组元素个数（常在循环语句中用到）

数组的拓展　数组函数

push（）函数

【格式】数组名.push（要在原数组中添加的新元素）

【功能】给原数组末尾添加新元素

【返回值】添加以后数组的元素个数

```
//例：    var abc =["蜘蛛侠","钢铁侠","绿巨人"];   //声明数组 abc 并且初始化
          Var def = abc . push ["蚁人","蝙蝠侠"];   //在原数组后面添加新元素
          alert(abc);          //警告框输出数组添加新元素后的所有元素
          alert(def);          //警告款输出数组添加新元素后的元素个数
```

pop（）函数

【格式】数组 .pop（）；

【功能】移除数组末尾的最后一位元素

【返回值】移除的元素

```
//例：    var abc = ["钢铁侠", "蚁人", "绿巨人"];   //声明数组 abc 并且初始化
          var def =abc .pop() ;        //摘除数组 abc 最末一位元素赋给变量 def
```

```
alert(abc) ;          //警告框输出数组摘除最末一位元素后的所有元素
alert(def);           //警告款输出原数组 abc 中的最末一位元素
```

shift（）函数

【格式】数组 .shift（）；
【功能】移除数组开头的第一位元素
【返回值】移除的元素

```
//例：    var abc = ["钢铁侠", "蚁人”, "绿巨人"];   //声明数组 abc 并且初始化
          var def =abc . shift() ;        //摘除数组 abc 开头第一位元素赋给变量 def
          alert(abc) ;          //警告框输出数组摘除开头一位元素后的所有元素
          alert(def);           //警告款输出原数组 abc 中的开头第一位元素
```

unshift（）函数

【格式】数组 .unshift（参数）；
【功能】从数组的开头插入元素
【参数】我们要插入数组的新元素，插入参数的个数随意
【返回值】添加以后数组的元素个数

```
//例：    var abc = ["钢铁侠", "蚁人”, "绿巨人"];   //声明数组 abc 并且初始化
          var def =abc . unshift("黑寡妇","蝙蝠侠") ;//在原数组开头插入的元素
          alert(abc) ;          //警告框输出数组添加新元素后的所有元素
          alert(def);           //警告款输出数组添加新元素后的元素个数
```

concat（）函数

【格式】数组 1 . concet（数组 2）；
【功能】将两个数组合并成一个新数组，源数组不会被改变
【返回值】合并好的新数组
【参数】我们要合并的数组

```
//例：    var abc = ["钢铁侠","绿巨人"];  //声明数组 abc 并且初始化
          var def = ["美国队长","鹰眼"];  //声明数组 def 并且初始化
          var ghi = abc . concat(def);          //合并数值 abc 与数组 def
          alert(ghi);           //警告框输出由数组 abc 及数组 def 合成的新数组 ghi
          alert(abc);           //警告框输出源数组 abc 内的所有元素
          alert(def);           //警告框输出源数组 def 内的所有元素
```

slice（）函数

【格式】数组 .slice(start, end);
【功能】基于当前数组获取指定区域元素并创建一个新数组。源数组不改变。
【参数】start 开始获取区域的下标，end 结束获取区域的下标，不包括 end 下标位置的元素。
【返回值】指定区域元素生成的新数组。

```
//例：    var abc=["蚁人", "鹰眼","超人”,"钢铁侠"];   //声明数组 abc 并且初始化
          var def = abc.slice(1, 3);     //截取数组 abc 中[1,3)的元素生成新数组 def
          alert(def);           //警告框输出新数组 def 内的所有元素
          alert(abc) ;          //警告框输出源数组 abc 内的所有元素
```

splice（）函数

【格式】数组 . splice(start,length,元素…);
【功能】可以完成删除，插入，替换操作
【参数】　参数 satrt：截取的开始下标
　　参数 length：截取的元素长度　//可以为 0
　　参数 元素…：要替换的新元素 //可以是多个元素
【返回值】截取的元素组成的新数组

```
//例：演示插入功能　删除、替换功能略
    var abc = ["鹰眼","蚁人","超人"];　　//声明数组 abc 并且初始化
    var def = abc . splice(1,0,"钢铁侠","蜘蛛侠");
    //在第一个元素"鹰眼"后面截取 0 个元素，替换为"钢铁侠""蜘蛛侠"
    alert(abc);　//警告框输出新数组 abc 内的所有元素
    alert(def);　//警告框输出新数组 def 内的所有元素
```

join（）函数

【格式】数组 . join(拼接符);
【功能】使用拼接符将数组中元素拼接成字符串。
【参数】拼接符　　[注]拼接符可以是任意字符
【返回值】拼接好的字符串

```
//例：　var abc=[10,20,30];　　//声明数组 abc 并且初始化
    var def = abc.join("+");
    //使用拼接符连接数组 abc 中的各个元素并赋给数组 def
    alert(def);　　//警告框输出新数组 def 内的所有元素
```

reverse（）函数

【格式】数组 . reverse ();
【功能】将数组中的元素逆向排序

```
//例：　var arr = [20,40,10,30];　　//声明数组 arr 并且初始化
    arr . reverse();　　//对数组 arr 进行逆序
    alert(arr);　　//警告框输出排序好的数组 arr
```

【注意】是排列顺序反转了，不是按从大到小排序

sort（）函数

【格式】数组 . sort ();
【功能】将数组中的元素按照升序排序
【注】sort 默认是按照字符串进行排序，即对比 Ascll 进行排序

```
//例：　var arr = [4,3,5,1,2];　//声明数组 arr 并且初始化
    arr.sort();　　//对数组 arr 进行升序排序
    alert(arr);　　//警告框输出排序好的数组 arr
```

Math 函数对象

Math.round 函数

【格式】Math.round(数值)
【功能】对数值进行四舍五入
【返回值】四舍五入后的结果

Math.random 函数

【格式】Math.random()
【功能】在 0~1 之间产生随机数
【返回值】产生的随机数

Math.max 函数

【格式】Math.max(数值 1,数值 2,数值 3，……)
【功能】提取出数据中的最大值
【返回值】最大值结果

Math.min 函数

【格式】Math.main(数值 1,数值 2,数值 3，……)
【功能】提取出数据中的最小值
【返回值】最小值结果

Math.abs 函数

【格式】Math.abs(数值)
【功能】求出数值的绝对值
【返回值】绝对值结果

Math.ceil 函数

【格式】Math.ceil(数值)
【功能】向上取整
【返回值】数值+1 再去除小数部分后的结果

Math.floor 函数

【格式】Math.floor(数值)
【功能】向下取整
【返回值】数值去除小数部分后的结果

Math.pow 函数

【格式】Math.pow(x,y)
【功能】求 x 的 y 次方的值
【参数】x 是底数，y 是指数
【返回值】x 的 y 次方的值

Math.sqrt 函数

【格式】Math.sqrt(数值)
【功能】求数值开平方的结果
【返回值】数值开平方后的结果

Math 对象勾股函数

Math.sin 函数
【格式】Math.sin(参数)
【参数】弧度数 ->例：n 弧度=n*Math.PI/180
【返回值】正弦函数的计算结果

Math.cos 函数
【格式】Math.cos(参数)
【参数】弧度数 ->例：n 弧度=n*Math.PI/180
【返回值】余弦函数的计算结果

Math.tan 函数
【格式】Math.tan(参数)
【参数】弧度数 ->例：n 弧度=n*Math.PI/180

其他 代码

严格模式

【使用格式】function m1{
"use strict"; <!-- 在想要使用严格模式的作用域下添加此段代码，作用范围在此作用域 -->
.....
}
【功能】在严格模式下，浏览器会对 js 的要求更加的苛刻

强制数据类型转换

强制数据类型转换：[Boolean()]

【使用格式】var 变量= Boolean(1)
【返回值】都将是 true 或者 false
【总结】数字 0 转换成布尔值为 false，所有非 0 的数字转换成布尔值都为 true
空字符串转成布尔值为 false，所有非空字符串转成布尔值为 true
特殊：null 和 nudefined 转成布尔值都是 false

强制数据类型转换：[Number()]

【功能】将别的数据类型转换为数字
【使用格式】var 变量= Boolean(1)
【总结】布尔值 true=>1 false=>0

字符串　纯数字的字符串=>对应的数字，否则 NaN

特殊数据类型　null=>0　　undefined=>NaN

强制数据类型转换：[Parseint()]

【功能】兼容 Number 功能（增加取整功能）

强制数据类型转换：[Parsefloat()]

【功能】兼容 Parseint 功能（可保留小数部分）

组合代码

数组遍历：

```
for(var　变量　in　数组)
{
循环体；
}
```

例

```
for(var　i　in　arr)
{
    document . write(arr(i)+"<br />");    //打印数组
}
```

其他　代码

换行　代码：

空格　代码：

符号替换代码：

"<"可以用&lt 替换；">"可以用&gt 替换。

全等(===)

(===)即数据类型、值全都相等

全不等（!==）

(!==)即数据类型、值全不相等

与（&&）

(表达式 1 && 表达式 2)

或（||）

(表达式 1 || 表达式 2)

非（！）

(! 表达式)

易错　问题

计算问题：

在 css 计算中被除数是可以为 0 的

例：var a =1/0;//计算结果为 infinity，意思为无穷大

var a =-1/0;//计算结果为-infinity，意思为无穷小

如果在声明变量的时候没有赋值，那么一般情况下将当前这个变量的值设置成 null

且运算高于或运算，如果一个语句中同时出现了且跟或，将先执行且语句再执行或语句