JS 笔记

JS 组成：
1.ECMAScript、DOM(页面文档对象模型)、BOM(浏览器对象模型)
例子：
<input type="button" value="“ onclick=”arert()"/>

2. onclick:点击事件；注：所有输出文本全部用单引号，方便阅读
<script> alert()</script> js 代码可以写到<script>标签中

3.prompt:输入框；（弹出警(提)示框）

4.Console:控制台
4.1：alert(msg)浏览器弹出警示框
4.2：console.log(msg) 浏览器控制台打印输出信息
4.3：prompt(info):浏览器弹出输入框，用户可以输入；

5.变量
作用：用于存放数据的容器，通过变量名获取数据；
本质：变量是程序在内申请的一块放数据的空间
使用：1.声明变量 ； 2.赋值；

5.1：声明变量 var 变量=' '； 用 console.log(变量）在控制台输出；
区别：javascript 是一种弱类型的动态脚本语言，其变量“var"作为数据类型，在其后面添加变量名后，在程序执行的过程当中
自动识别变量名的类型；
例如：var x=10; //x 是数字类型 x='pink'; //x 是字符串类型 （var 可以连续赋值，但是前面先赋值的数据会被后面的数据所
覆盖，并且数值类型也相继的发生改变）；

5.2：多个变量声明：var 变量 1=''，变量 2=''，.....;

5.3 变量命名规范：
5.3.1(命名组成)：
1.命名组成是由 字母（A-Z,a-z)、数字（0-9)、下划线" _"、美元符号($)组成；
2.严格区分大小写，var app 和 var App 是两个变量。
3.不能以数字开头。
4.不能是系统关键字、保留字。
5.变量命名要有实际意义。
6.驼峰命名法：首字母小写，后面单词的首字母都需要大写。

6.数据类型简介
6.1：JS 把数据类型分成两类：1.简单数据类型(Number、String、Boolean、Undefine、

Null)；2.复杂数据类型(object)；

6.2:进制转换：

八进制：0~7 表示：var num1=010; 其中 010 就是八进制数

十六进制：0~9,a~f; var num=0x....; 其中带 0x 就是十六进制数

7.数字型范围

7.1：javascript 中数值的最小值和最大值；最大值：alert(Number.MAX_VALUE);显示为：1.7976.....e+308;最小值：alert(Number.MIN_VALUE);显示为：5e-304

7.2:alert(Infinity);无穷大值 ； alert(-Infinity);无穷小值； NaN ,Not a number,代表一个非数值。

7.3：isNaN(),作用：判断一个变量是否为非数字的类型，返回 true 与 flase；x 是数字，返回 false，不是数字，返回 true；

7.4 typeof:显示类型；

7.5:toString:转换为字符串型

例如：1.把数字型转换为字符串型，变量.toString();

```
var num=10;
var str=num.toString();
console.log(typeof str);
```

2.利用 String(变量)

```
console.log(String(num));
```

3.利用"+“拼接字符串的方法来实现转换效果

```
console.log(num+'');
```

7.6:parseInt(变量)

作用：把字字符型的转换为数字型，得到的是整数

例：

```
console.log(parseInt('3.14'));        //3 取整
console.log(parseInt('3.94'));        //3 取整
console.log(parseInt('120px'));    //120 会去得到这个 px 单位。
console.log(parseInt('rem120px')); NaN
```

7.7:parseFloat(变量)

作用：可以把字符型的转换为数字性，得到的是小数。

例：

```
console.log(parseFloat('3.14'));    //3.14
console.log(parseFloat('120px'));    //120 会去得到这个 px 单位
console.log(parseFloat('rem120px');   NaN
```

8.字符串引号嵌套

8.1： 内容：JS 可以单引号嵌套双引号，或者双引号嵌套单引号；

例如：var str='我是“。。。"学生'; 或者 var str="我是'。。。'学生“;

8.2：\n :换行符， \\ :斜杠“ \”， \' 单引号:“ ’”， 双引号 \": " " " ， \t:tab 缩进 ，\b:空格，b 是 blank 的意思。

8.3：字符串长度：字符串是由若干个字符组成，这些字符数量就是字符串的长度。通过字符串 length 属性可以获取整个字符串的长度。

例如：var str=“123456789”;alert(str.length); //显示为字符串的长度：9；

9.字符串拼接

console.log('沙漠'+'骆驼');输出”沙漠骆驼“；

console.log(12+12) 或者 console.log('12'+12)，前者因为都是数值类型，所以它们直接进行相加计算，显示为”24“，后者因为'12'是字符串类型，而”12“是数值类型，类型取”+“前面的类型进行显示，最后显示为”1212“；

10.递增和递减运算符的概述

概念：如果反复给数字的变量添加或减去 1，可以使用递增(++)或者递减(--)运算符来完成。

注：递增或递减运算符必须配合变量来使用。

11.算数运算符

11.1：”==“算数符（判断 两边的值是否相等）

作用：在程序里面就是”等于“符号，是”==“默认转换数据类型 会把字符串的数据转换为数值型 只要求相等就可以

console.log(3==5); false; 返回为 boolean 类型

console.log(18==18) ;true

console.log(18!=18) ;false

11.2:"==="算数符“（判断两边的值和数据类型是否完全相同）

作用：在程序里表示为”全等 ”、“一模一样”，它要求的是"数据类型"和"数值"必须全部一致；

例：

console.log(18===18); true

console.log(18==='18') false 虽然它们数值相等，但是全等是要求数值类型和数值都必须相等，所以返回值为 false；

12：数组

概念：数组是指一组数据的结合，其中的每一个数据被称作为“元素”，在数组当中可以存放任意类型的元素。

例：普通变量只能储存一个值" var num=10; " 数组一次性可以存储多个值 " var arr=[1,2,3,4,5];"

注：当索引下标大于了数组本来的大小，那么索引的值为“Undefined";

12.1:创建数组

12.1.1：数组的创建方式：

1.利用 new 创建数组；

例：var 数组名=new Array(); //创建一

个新的空数组　"A"注意大写

2.利用数组字面量来创建数组；

例：1.　var arr=[];//创建一个空的数组

2.　var arr1=[1,2,'学生'，true];

注：数组里面的元素一定用逗号分隔

12.2:数组长度索引

例：数组名.length;

var arr=[];　arr.length;//作用：返回 arr 数组的大小

注：数组的长度是数组所包含的元素的总个数;

12.3：标识符命名规范

1. 变量、函数的命名必须有意义;
2. 变量的名称一般用名词;
3. 函数的名称一般用动词;

13：函数

13.1：函数声明:

13.1.1:函数声明有两步："声明函数"和"调用函数"

例：声明函数

```
function 函数名（）{
        函数体
}
function sayHi(){
        console.log('hi~);
}
```

注：1.function 声明函数的关键字 全部小写；2.函数是做某一件事情，函数名一般是动词；3.函数命名不用带"var",直接写形参；（但是对于强类型语言，例如"java"、"c,c++"等语言在声明函数时，要么函数参数为空，要么形参一定是有对应的类型）

13.2：调用函数:

注:声明函数本身并不会执行代码，只有调用函数才会执行函数体代码。

调用方式：函数名（）；//调用函数的时候不要忘了加括号

口诀：函数不调用，自己不执行;

13.3：函数形参个数的匹配:

例：

```
function getSum(num1,num2){
    console.log(num1+num2);
}
```

1.如果实参的个数和形参的个数一致，则会输出结果

```
getSum(1,2);
```

2.如果实参的个数多于形参的个数，会取到形参的个数

getSum(1,2,3);

3.如果实参的个数小于形参的个数 多余的形参定义为 undefined;

形参可以看作是不用声明的变量 num2 是一个变量但是没有接收值，那么输出结果是 NaN;

getSum(1);

13.4：函数"return"注意事项：

13.4.1：return 终止函数：

```
function getSum(num1,num2) {
    return num1+num2;  //return 后面的代码都不会执行
    alert('我是不会执行的');
}
```

13.4.2: 返回数组类型:

```
function getResult(num1,num2) {
    return [num1+num2,num1-num2,num1*num2];
}
var re=getResult(1,2); //返回的是一个数组
console.log(re);
```

13.5:argunments 的使用：

概念：当我们不确定多少个参数传递的时候，可以用”arguments“来获取，在 javascript 中，arguments 实际上它是当前函数的一个内置“对象”。所有函数都内置个“arguments”对象，"arguments"存储了传递的所有实参;

使用注意："arguments" 的使用 ：只有函数才有”arguments“对象，而且每个函数都内置了”arguments"；并且，它被称为“伪数组”，并不是真正意义上的数组;

“arguments"特点:

1.具有数组的 length 属性;

2.按照数组下标索引的方式进行储存;

3.它没有真正数组的一些方法 pop(),push();

14.javaScript 作用域：

概念：就是代码名字（变量）在某个区域内起作用和效果 目的就是为了提高程序的高效性和减少命名的冲突;

14.2：局部作用域：

概念：局部作用域（函数作用域）在函数内部就是局部作用域，这个代码的名字只在函数内部起效果作用；

例：

```
function fn() {
    var num=20;
    console.log(num);
}
fn();
```

14.3:变量的作用域：

概念：根据作用域的不同将变量分为"全局变量"和"局部变量";

注意：如果在函数内部，没有声明就直接赋值的变量也属于"全局变量";

例：

```
var num=10; //num 就是一个全局变量，因为它在Script 标签内，在函数外，作用于整个 Script 标签内
console.log(num);

function fn() {
    console.log(num);
}

fn();

//2.局部变量 （在局部作用域下的变量 "在函数内部的变量"）
function fun() {
    var num1=10; // num 就是局部变量 只能在函数内部使用
    num2=20;  //num2 为全局变量，因为它没有声明就直接赋值"20";
}
fun();
console.log(num2);
```

14.4:全局变量与局部变量的特点：

1.全局变量：只有浏览器关闭的时候才会销毁数据，所以比较占资源；

2.局部变量：当程序执行完毕就会自动销毁，所有节约空间资源；

14.5：作用域链

1.只要是代码，就至少有一个作用域；

2.写在函数内部的局部作用域；

3.如果函数中还有函数，那么在这个作用域中就可以重新诞生一个作用域；

4.根据在函数内部可以访问外部函数变量的这种机制，用链式查找决定哪些数据能被内部函数访问，就称作为作用域链；

14.6：预解析

概念：　　预解析运行 js 代码分为两步：1.预解析；2.代码执行；

14.6.1：预解析 js 引擎会把 js 里面所有的“var”还有“function”提升到当前作用域最前面

14.6.2：代码执行会按照代码书写的顺序从上往下依次执行；

14.6.3：预解析分为“变量预解析(变量提升)和”函数预解析“（函数提升）；

14.6.4：变量提升就是把所有的变量声明提升到当前作用域最前面 ，但是不会提升赋值操作；函数提升，就是把所有的函数声明提升到当前作用域的最前面，不用调函数

例：

```
function f1() {
    var a;

a=b=c=9; //相当于 var =9；b=9; c=9;b 和 c 直接就赋值了，所以"b"和“c"直接可以看作是全局
//变量，因为它们两没有"var”声明;


console.log(a);

console.log(b);

console.log(c);
}
f1();

console.log(c);

console.log(b);

console.log(a);
```

15.对象

概念：在 JavaScript 中，对象是一组无序的相关“属性”和“方法”的集合，所有的事物都是对象，例如字符串、数组，数值、函数等；

15.2：创建对象的三种方式：

1.利用“字面量”创建对象；
2.利用“new Object"创建对象；
3.利用 "构造函数”创建对象；

15.3：调用对象的方法：

对象名.方法名() 注：别忘记加括号

obj.SayHi();

1.变量 单独声明并赋值 使用的时候直接写变量名 单独存在；
2.属性 在对象里面的不需要声明的 使用的时候必须是 "对象.属性“
3.函数和方法的相同点：都是实现某种功能 做某件事；方法 在对象里面调用的时候 对象.方法();

例：

利用 new Object 创建对象

```
var obj=new Object(); //创建了一个空的对象
obj.uname='张三';
obj.age=18;
obj.sex='男';
obj.sayHi=function() {
        console.log('hi~);
    }
        //1.利用”等号=赋值的方法“添加对象的属性和方法
        //2.每个属性和方法之间用”分号“结束
console.log(obj.uname);
console.log(obj['sex']);
obj.sayHi();
```

15.4：构造函数

概念：在创建一个对象时，里面有很多的属性和方法是大量相同的，就只能复制，所以就可以利用构造函数的方法，重复这些相同的代码，我们把这个函数称作为”构造函数"；但又因为这个函数不一样，里面封装的不是普通的代码，而是"对象“。构造函数就是把我们对象里面一些相同的属性和方法抽象出来的封装到函数里面；

例：

```
// function 构造函数名 () {
//          this.属性=值;
//          this.方法=function () {}
//      }
```

// new 构造函数名();

注：1.构造函数名字的首字母"大写" 2.我们构造函数不需要"return"就可以返回结果 3.调用构造函数必须使用"new"；

15.5："for...in"

作用："for in"遍历我们的对象

for(变量 in 对象){}

for（var k in obj){

console.log(k); //k 变量输出 得到的是"属性名"

console.log(obj[k]); // obj[k] 得到的是 属性值

}

15.6:"indexOf"和 "lastIndexof()"

概念："indexOf"是返回查询数组元素下标；从前往后；"lastIndexOf"也是返回查询数组元素下标，"从后往前"；

它们只返回第一个满足条件的索引号，如果在该数组里面找不到元素，就返回"-1"；

例：arr.indexOf('查找元素','开始下标'), arr.lastIndexOf('查找元素','开始下标');

var =['red','green','pink'];

console.log(arr.indexOf('blue')); //返回数组元素

15.7：数组转换为字符串

1.toString() //将数组转换为字符串

2.join(分隔符)

var arr1=['green','blue','pink'];

console.log(arr1.join()); //green,blue,pink;

console.log(arr1.join('-'))//green-blue-pink;

15.8: "replace"替换

例：arr.replace('当前数组元素','替换元素');

16.内置对象

概念：javascript 中的对象分为 3 种：自定义对象、内置对象、浏览器对象，"内置对象"就是指 JS 语言自带的一些对象，这些对象供我们使用，并且提供了一些常用的或是最基本的必要的功能(属性和方法);

16.1:Math 概述：

概念：Math 对象不是构造函数，它具有数学常数和属性和方法。(求绝对值、取整、最大值等) 可以使用 Math 中的成员。

Math.PI //圆周率 Math.floor（）//向下取整

Math.ceil()//向上取整　　Math.round() //四舍五入　　Math.abs（）//绝对值

16.2：Date()方法的使用

1.获取当前时间必须实例化

var now=new Date（）;

console.log（now）;

注：如果括号里面有时间，就返回参数里面的时间。例如日期格式字符串为'2020-5-1',可以写成 new Date（'2020-5-1')或者 new Date('2020/5/1）

16.2.1：var getTime=new Date(); //注：Date 的“D”记得大写

理解：要求返回一个当前的一个系统时间，用”getTime"来调用一系列的方法，例如“getime.getFullYear()"、getTime.getHours()”等方法调用；

16.2.2：var time=+new Date(“放时间【也可以不放，不自定义放时间就是返回当前系统时间的”总毫秒数“】”);

理解：在 对象前面加“+”就是要求系统返回当前的“总毫秒数”;

16.3:调用对象

概念:

1. 对象里面的属性或者方法我们采取“键值对”的形式 "键“：属性名 值：属性值

2.多个属性或者方法中间用“，”隔开;

3.方法冒号“：”后面更着一个匿名函数

调用方法:

1.调用对象的属性 ，我们采取 “对象名”.“属性名" ,"."理解为调用

2.调用还有一种方法：”对象名【'属性名'】“　　;

16.4：查看器 "instanceof;

概念：”instanceof"是查看当前对象是为哪一种类型

使用方法:

例：var arr=[];

var obj={};

console.log(arr instanceof Array); //arr 是一个数组，所以为"true"

console.log(obj instanceof Array);//obj 不是一个数组，所以为“false";

2.Array.isArray(参数) //H5 新增的方法

console.log(Array.isArray(arr););

console.log(Array.isArray(obj););

2021-11-23　√

17.API

概念：“API,应用程序编程接口”，是一些预先定义的函数，提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力。

17.1:根据标签名获取

使用 getElementsByTagName()方法可以返回带有指定标签名的对象的集合。

17.2:通过 HTML5 新增的方法获取

1.doucument.getElementsByClassName('类名') //根据类名返回元素对象集合

2.doucument.querySelector('选择器'); //根据指定选择器返回第一个元素对象

3.doucument.querySelectorAll('选择器') //根据指定选择器返回

17.3：获取特殊元素

1.获取 body 元素

doucunment.body //返回 body 元素对象

2.获取 html 元素

doucument. doucumentElemment //返回 html 元素对象

17.4：样式属性操作

概念：可以通过 JS 修改元素的大小、颜色、位置等样式。

1.element.style //行内样式操作

2.element.className //类名样式操作

注意：

1.如果样式修改较多，可以采取操作类名方式更改元素样式。

2.class 因为是保留字,因此使用 className 来操作元素类名属性。

3.className 会直接更改元素的类名，会覆盖原先的类名。

17.5：事件触发类型

概念：事件是由三部分组成:事件源 事件类型 事件处理程序 ，这三部分合并称为事件三要素

1.事件源：事件被触发的对象

2.事件类型：如何触发 什么事件 例如鼠标点击(onclick) 还是鼠标经过 还是键盘按下

3.事件处理程序 通过一个函数赋值的方式 完成;

例如:var btn=document.getElementById('对象');

btn.onclick=function(){ 事件处理 }

17.5:"innerText"与“innerHTML"的区别

1.innerText 它不识别 html 标签 非标准 它会去除空格和换行

2.innerHTML 识别 html 标签 W3C 标准 保留空格和换行

17.6：失去焦点，获取焦点

获取焦点：对象.onfocus=function(){}

失去焦点：对象 onblur=function(){}

2021-11-25

17.7：onmouseover(鼠标经过)、onmouseout(鼠标离开)

17.8：操作元素：

1.自定义属性的操作

element.属性 获得属性值(元素本身自带的属性)

element.getAttribute('属性');(获取的是自定义的属性(程序员命名))；

17.9：设置 H5 自定义属性

H5 规定自定义属性 data-开头做为属性名并且赋值。

例如：

17.10：节点概述

概念：节点至少拥有 nodeType(节点类型)、nodeName(节点名称)和 nodeValue(节点值)三个基本属性

1.元素节点：nodeType 为 1；

2.属性节点：nodeType 为 2；

3.文本节点：nodeType 为 3；(文本节点包含文字、空格、换行);

17.11:节点操作

1.节点层级： 利用 DOM 树把节点划分不同的层级关系，常见为"父子兄层级关系";

2.父级节点：

node.parentNode;

概念：parentNode 属性可以返回某一节点的父级节点，注意是"最近的一个父级节点";

如果指定的节点没有父级节点那么就返回"null”;

17.12:节点操作:

1.子节点：parentNode.children （非标准)

概念：parentNode.children 是一个只读属性，返回所有子元素节点，它只“返回子元素节点，其余节点不返回”；

2.parentNode.fristElementChild

fristElementChild 返回第一个子元素节点，找不到就返回 null；

3.parentNode.lastElementChlid

lastElementChlid 返回最后一个子元素节点，找不到就返回 null;