# 快捷键：

ctrl + / 单行注释： //

alt + shift + a 多行注释 ： /\* \*/

Ctrl + F 搜索

# JavaScript:入门

* JavaScript是运行在浏览器端的脚步语言，JavaScript主要解决的是前端与用户交互的问题，包括使用交互与数据交互。 JavaScript是浏览器解释执行的，前端脚本语言还有JScript（微软，IE独有），ActionScript( Adobe公司，需要插件)等。
* **前端三大块:**

1、HTML：页面结构

2、CSS：页面表现：元素大小、颜色、位置、隐藏或显示、部分动画效果

3、JavaScript：页面行为：部分动画效果、页面与用户的交互、页面功能

## 1.1 js的三种书写方式

* 行内式
* 嵌入式
* 外链式
  + 注意点：如果一个 script标签 链接了一个 js文件, 这个script 就不能再做其他事情

1. 行内式：

<!-- 第一种JavaScript插入页面的方式: 行内式/行间事件 不推荐 -->

<div style="color:red" onclick="alert('hello world!')">这是一个div</div>

2．嵌入式：

<!-- 第二种JavaScript插入页面的方式:内嵌式 部分推荐 -->

<script>

alert("内嵌式"); // 弹出警告框

</script>

3.外链式：

<!-- 第三种JavaScript插入页面的三种方式:外链式 推荐 -->

<script src="./js/hello.js"></script>

## 1.2 变量的定义：

* 变量可以单个定义，也可以一次定义多个，但要用逗号隔开。

1.单个定义：

var iNum = 12; 数字

var sTr = 'abc'; 字符串

var bIsMove = false; 布尔型

2. 多个定义：

var iNum = 12, sTr = 'abc', bIsMove = true;

## 1.3 变量的类型

### 1.3.1 基本数据类型

* 数字类型 int float
* 字符串 string
* 布尔类型 true false
* 未定义 undefined 变量未赋值就是undefined
* 空类型 null

// 变量声明,但不赋值,它的值和类型就是undefined

var iNum02;

// 修改变量的值

iNum = 15;

### 复合类型 object：

复合类型：

<script>

var oObj = {

name:"张三",

age:16,

}

alert(oObj.age) // 获取对象 age值

</script>

* typeof 获取对象类型：

typeof (oObj.name)

### 1.3.3 命名规范：

1、 严格区分大小写

2、 首字符 : 字母 下划线(\_) $

3、 除了首字符之外的其他字符: 数字 字母 下划线 $

* **匈牙利命名风格：**
* 对象o Object 比如：oDiv
* 数组a Array 比如：aItems
* 字符串s String 比如：sUserName
* 整数i Integer 比如：iItemCount
* 布尔值b Boolean 比如：bIsComplete
* 浮点数f Float 比如：fPrice
* 函数fn Function 比如：fnHandler
* 正则表达式re RegExp 比如：reEmailCheck

## 1.4 函数的定义：

**关键字 function**

1.无参数的函数定义:

function fnTest() {

alert('hello');

}

fnTest(); // 函数调用/执行

2.有参数的函数定义:

function fnResult(a,b) {

// alert(a + b);

// return关键字的作用:返回函数中的值或对象,也可以结束函数的运行

return a + b;

}

var iResult = fnResult(1,2);

alert(iResult);

3.return关键字的作用:

(1) 返回函数中的值或对象

(2) 结束函数的运行

## 1.5 函数/变量预解析：

1.变量的预解析:

如果前面先使用变量,后面写变量的声明,此时变量值为undefined（拿过去的是声明而不是声明和值）

2.函数的预解析:

如果前面先写执行,后面写函数定义,系统会在执行的 检查代码,能正常调用

<script>

1.函数的预解析:函数在后面定义,在前面正常调用

fnTest();

function fnTest() {

alert("see you aglin!");

}

2、变量的预解析:变量在后面定义,在前面使用,变量的值为undefined

alert(iNum);

var iNum = 12;

</script>

## 1.6 条件语句：

1. "==" js中会默认转换数据类型,将数据类型转换成统一类型后再比较

2. "===" 不会转换数据类型,如果类型不一样就是不相等了

3. "&&" 一假则假,只要有一个条件不成立那就不成立

4. "||" 一真则真,只要有一个条件成立那就为真

5. "!" 取反 真变假 假变真

6. “！= “ 不等于

7. “！== “ 不全等

《基本语法》

if (条件1) {

} else if (条件2){

} else {

}

## 1.7 获取元素（标签）

1. 定义在body最底部 （ 项目中常用）：

<body>

<div id="div1">这是一个div元素</div>

<script>

// id 值一般用 单引号 ‘’

var oDiv = document.getElementById('div1'); // 获取标签对象

</script>

</body>

1. 在 head 里获取 ：

<script>

// 界面加载完成事件：包括HTML 代码，以及外部资源加载完成

// 当整个页面加载完之后,再执行大括号里面的代码

window.onload = function () { // on… 事件

// 通过id获取获取元素

var oDiv = document.getElementById('div1');

// 操作元素的样式属性 CSS

oDiv.style.color = 'red';

oDiv.style.width = '200px';

oDiv.style.height = '200px';

oDiv.style.backgroundColor = 'gold';

// js中操作css属性时属性名中用 - 时要去掉并把后面的转换成驼峰命名法

oDiv.style.fontSize = '30px';

}

</script>

## 1.8 读写元素（标签）属性：

1、js中使用css样式属性是，把属性中间的 - 去掉 后面的单词首字母大写

2、修改标签里的内容：

innerTxt ： 以纯文本传入

innerHTML ： 文本 和 标签

var oDiv = document.getElementById('div1');

var oA = document.getElementById('link01');

// 写入属性,也叫设置属性

oDiv.style.color = 'red';

oA.href = 'http://www.itcast.cn';

// 修改类名来更换样式

oA.className = 'a02';

// 读取标签里包裹的内容

alert(oDiv.innerHTML);

alert(oA.innerHTML);

// 修改标签内包裹的内容

oA.innerHTML = '传智播客';

oDiv.innerHTML = '<a href="#" id="link01" class="a01">个链接</a>';

## 1.9 匿名函数：

如果此函数只用一次,而且还是触发事件后才执行的函数就可以简化为匿名函数'没有名字的函数'

1、单独的匿名函数 报错

2、匿名函数必须要赋值给变量或属性

3、onclick 监听按钮点击事件，当按钮点击时调用方法

## 1.10 案例：

### 网页换肤案例：

1、修改link标签的href属性：（<link rel="stylesheet" href="./css/skin01.css" id="link01">）

<script>

var oLink = document.getElementById('link01')

oBtn01.onclick = function () {

oLink.href = 'css/skin01.css'; // 修改外链的路径值

}

</script>

### 打印名片：

// parseInt() 转换为int类型

// 更换class来修改样式， 拼接字符串

var sClassName = 'idcard0' + (parseInt(oInput07.value) + 1);

oCard\_wrap.className = sClassName;

js获取class属性: .classname

+ : 加号左右两侧如果有字符串,会将另一侧的数字转换为字符串. 进行字符串拼接

+ : 如果两侧都是数字, 数字相加

# 2. JavaScript:高级

## 2.1 数组的定义：

数组变量:存储多个数据

普通变量:存储一个数据

1. 创建数组对象：实例化对象 (new:创建)

var oList01 = new Array(1,2,3);

1. 快速创建：

(注意: js是弱类型语言.允许数组中的元素的数据类型不一致. 但是一般我们定义的时候,都是一致的)

var oList02 = ['a','b','c',1,'a']; //常用

// console.log(oList01, oList02); // 控制台输出

1. 多维数组：(数组中的元素/项,还是一个数组)

var aThreeArray = [[1,2,3],["a","b","c"]];

## 2.2 数组的操作：

1. length长度 用来获取数组的长度,元素个数 (length属性) \*\*\*

console.log(aList01.length);

2. push添加 在数组最后面添加新的元素 (数组的元素:数组中某一项) (push推)

aList01.push(6);

push()里面如果传多个参数表示一次给数组添加多个元素

aList01.push(6,7);

3. pop删除 数组中最后一个元素,一次只能删除最后面的一个 (pop:爆炸)

aList01.pop();

4 .获取 数组中指定角标/下标/索引的元素 \*\*\*\*

var iNum = aList01[4];

5.修改 数组中指定角标的元素 \*\*\*\*

aList01[0] = '1';

6. splice : 删除/替换/添加 (有点难) (splice:胶接)

    第一个参数: start 表示要删除的元素起始角标,包含起始角标

第二个参数: deleteCount 表示要删除的元素个数

第三个及以后的参数: newElement 是要添加的新元素,添加在第一个参数的角标位置起,占有起始角标

             var aList01 = [1, 2, 3, 4, 5];

       1. 删除

             aList01.splice(0,2); // [ 3,4,5 ]

       2. 替换

            aList01.splice(0, 3, '1', '2'); // [ ‘1’,’2’,4,5 ]

       3. 添加

           aList01.splice(0, 0, '1', '2'); // ['1', '2', 1, 2, 3, 4, 5]

7. reverse : 反转 :把数组中的每一个元素颠倒/倒置/反转 \*\*\*

aList01.reverse();

8. join : 拼接 : 把数组中的元素 ,以某个字符 进行拼接 变成字符串 \*\*\*

        var aList01 = [1, 2, 3, 4, 5];

        var sStr = aList01.join(); // 1,2,3,4,5 (默认按照数组的逗号拼接)

        var sStr = aList01.join(''); //123456

        var sStr = aList01.join('-'); //1-2-3-4-5-6

   注意:

            数组与字符串 , 相互转换. (此逻辑,开发中用的比较多)

9. indexOf : 谁的索引: 如果找到了,就是对应索引(只找第一个). 没有找到,就是-1; \*\*\*

     返回元素在数组中第一次出现的角标

// 如果返回-1说明数组中没有些元素

var aList02 = ['1', '2', '1'];

         console.log(aList02.indexOf('z'));

         注意:

             判断某个元素是否在数组中.

                如果等于-1,说明不存在 ==-1

                如果不等于-1,说明存在    !=-1

### 2.2.1 数组去重：

#### 方法1：

（1）遍历数组有序的取出每一个元素，然后再取出元素在数组中第一次出现的角标，

（2）判断当前遍历的个数索引和元素在数组中的第一次出现的角标是否一样，相同就是第一次出现，然后添加到新的数组

（3）不相同就不添加说明是重复的

<script>

var oList = ['a','b','a'];

var oList2 = [];

for (var i = 0; i < oList.length; i++) {

// 获取数组中指定角标的元素

var element = oList[i];

// 获取元素在数组中第一次出现的角标

var iIndex = oList.indexOf(element);

// 如果当前循环的索引和元素在数组中第一次出现的角标一样,说明此元素是第一次出现

if (i == iIndex) {

// 第一次出现的元素添加到数组中

oList2.push(element);

}

}

console.log(oList2);

</script>

#### 方法2：

1. 取出数组中的每一项

2. 判断当前项是否在新数组中,如果没有,就放入

    for (var i = 0; i < aList01.length; i++) {

         // 取出数组中的每一项

        var sItem = aList01[i];

         // 判断当前项是否在新数组中,如果没有,就放入

        if(aNewList.indexOf(sItem) == -1 ){

            aNewList.push(sItem);

        }

    }

## 2.3 循环语句：

1. while循环

var iNum01 = 0;

while (iNum01 < 5) {

iNum01++;

console.log(iNum01);

}

2. for循环 \*\*\*

for (var i = 0; i < 5; i++) {

console.log('哈哈' + i);

}

执行流程: \*\*\*

1. 初始化表达式 : var i = 0; (只执行一次)

2. 条件表达式 : i < 5

3. 循环体: console.log('哈哈' + i);

4. 叠加/递减表达式 : i++

另外的一种写法:

var i = 0;

for ( ; i < 5; ) { // 这里的;不能省去

console.log('哈哈' + i);

i++;

}

var iNum01 = 0;

while (iNum01 < 5) {

iNum01++;

console.log(iNum01);

}

3. 区别： 所有python中的条件不带小括号(). js中的条件都放在()中

if条件

while条件

for条件

## 2.4 列表加载数组数据：

1. 定义数组

2. 循环遍历数组

        拼接 列表 字符串

        str += "<li>"+arr[i]+"</li>"

3. 将拼接的字符串,添加到界面ul上

        ul.innerHTML = str;

## 2.5 字符串：

1. 类型转换 (字符串转换数值) \*\*\*

        parseInt 转换为整型 (integer)

        parseFloat 转换为float 浮点型

     浮点型: 某些情况下,浮点数做运算时丢失精度. 0.1 + 0.2 != 0.3

        - \* / == 隐式转换 系统自动判断 并转换类型

             15 == '15'

             NaN 非数值类型 : not a number

    做字符串 转换为 数值 类型时, 如果是非数字的内容进行转换. 转换结果就是NaN

                123 - 'abc'

2. 字符串的常用方法 \*\*\*

   （1） “ + ” 可以进行字符串拼接：

            + : 如果加号左右两侧,有一个字符串. 字符相连

               如果两侧都是数字,数字相加.

   （2） subString():字符串截取：

            参数: subString(start,end);

                 [start,end)包含start,不包含end, 左闭右开

                 如果只传入第一个参数,默认截取到最后

   （3）split() : 字符串拆分 :把字符串中的字符拆分成数组中的每一个元素

          var sStr = "abc";

1. 如果什么也不传就是把字符串当成数组中的一个元素

          sStr.split() -----🡪 ["abc"]

2. 如果传空字符串就是把字符串中的每一个字符拆分成数组的一个一个的元素

sStr.split('') ------🡪 [a,b,c]

3. 如果传入指定字符 表示以此字符来拆分字符串,如果字符串中没有此字符,

          var sStr03 = '2017-11-28';

sStr03.split('-') ----🡪 [2017,11,28]

4.indexOf() : 获取字符/字符串的索引 : 获取字符在字符串中第一次出现的角标或判断字符串中是否包含指定字符

            如果返回-1 表示字符串中没有此字符

            字符串: 也称字符数组.

                    var sStr01 = 'abcdefab', // sStr01[0] --- a

                    sStr01.indexOf('ab')

5.字符串与数组相互转换

            1.数组转换为字符串 : join

            2.字符串转换为数组 : split

## 2.6 倒置字符串：

1、split("") 字符串拆分成数组

2、reverse() 数组中的元素倒置/颠倒

3、join("") 数组转换成字符串/把数组中的元素连接成字符串

**例：**

var sStr11 = 'abcdef';

var sNowStr = sStr11.split('').reverse().join(''); // 拆分 – 反转 – 拼接

console.log(sNowStr);

## 2.7 调试方法：

* alert() 可以阻止程序的运行
* console.log() 控制器输出
* document.title = xxxx ; 更改网页窗口上的标题（在标题上输出）
* document.write("xxx") ; body中直接写入内容

## 2.8 全局和局部变量：

（1）正常情况局部变量的优先级比全局的高

（2）但在js中如果全局和局部变量重名时，局部变量定义的 大{}中 会直接屏蔽全局变量，所以在定义局部变量之前使用会出现undefined;

（3）尽量避免全局变量和局部变量同名

## 2.9 定时器：

参数：（ 功能,时间(毫秒) ）

（1）只执行一次的定时器：

// 第一个参数 是定时器要执行的函数 第二个参数 间隔时间 单位 毫秒

var oTimer = setTimeout(fnTest,2000);

// 停止/销毁只执行一次的定时器

clearTimeout(oTimer);

（2）重复执行的定时器：

// 第一个参数 是定时器要执行的函数 第二个参数 间隔时间 单位 毫秒

var oTimer01 = setInterval(fnTest,2000);

// 停止/销毁重复执行的定时器

clearInterval(oTimer01);

## 2.10 鼠标事件：

onmouseover 鼠标停留

onmouseout 鼠标离开

例：

oDiv.onmouseover = function () {

      clearInterval(oTimer); // 鼠标停留时，清除定时器

  }

oDiv.onmouseout = function () {

     oTimer = setInterval(fnMove,30); // 鼠标离开， 设置延时

  }

获取标签的内容，并叠加。

oUl.innerHTML += oUl.innerHTML;