**Javascript**

JS文件不能直接运行，需嵌入到HTML中执行，在HTML中添加如下代码，就可将JS文件嵌入HTML文件中。

<script type="text/javascript" src=""></script>

JavaScript注释作用：

1. 去除多余代码或临时禁用某段代码；
2. 对代码块的解释阐述，方便后续开发时快速理解代码含义及作用。

声明变量通过var来声明（变量名variable的前三个字母）

命名规范

代码中变量的命名规范：

可以包含字母[a-z]、数字[0-9]、下划线[ \_ ]、美元符号[$]

避免数字开头

名称最好有明确的含义，<div>→name、mingzi

匈牙利：mz 、下划线ming\_zi（）、小驼峰mingZi（第一个单词首字母小写，其余首字母大写）、大驼峰MingZi（每个单词的第一个字母都大写）

—可以采用常规命名法：“匈牙利命名法”、“下划线命名法”、“小驼峰命名法”、“大驼峰命名法”之一，在开发团队内进行协调统一

**JS的三种弹框:**

1. alert（）；②confirm（）；③prompt (“输入框+确认取消”)

**JS三种输出方式**：

1：window.alert();

2 : console.log() / console.warn() /console.error;

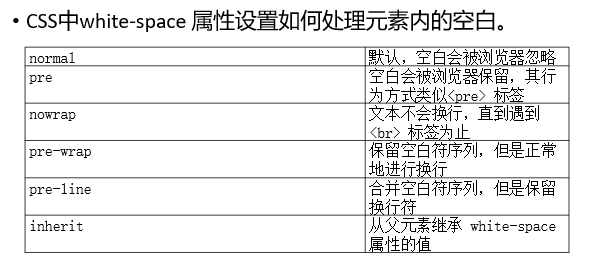
3：document.write();直接在页面输出内容（会覆盖原有内容）

**浏览器读取显示机制：**

如何保留空格 1：使用输出HTML标签…&nbsp；来解决；

2：使用css样式解决，

document.write(“<span style=‘white-space：pre;’>”+”1 2 3”+”</span>”);



**文档加载事件**

window.onload=function(){ };

表示页面加载完毕后执行

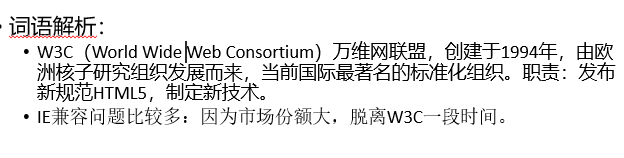
**DOM**

DOM是document object model，即文档对象模型，定义访问和处理HTML文档/元素的标准方法。

简单理解： DOM 是关于如何获取、修改、添加或删除 HTML 元素的标准，通过DOM可以访问HTML文档的所有元素。

当浏览器载入HTML时，即当网页被加载时，浏览器会创建页面的文档对象模型（Document Object Model）。DOM 将HTML文档呈现为带有元素、属性和文本的树结构（节点树），会生成相应的DOM树。在DOM 中，所有事物都是节点。DOM 是被视为节点树的 HTML。

所以节点分很多种。常见的有文档节点（整个文档）、元素节点（HTML是由元素构成，HTML的元素）、文本节点（text文本信息）、属性节点（href、title、alt）、注释节点（注释内容）等。





获取元素节点

1，document.getElementById("id");

2, document.getElementsByTagname("tagname")[下标]；

3，document.getElementsByClassname("classname")[下标]；

输出当前节点文本内容

console.log(x.innerHTML);

1. inner HTML：输出当前节点文本内容，连同子元素及子元素节点标签一同输出。
2. innertext：输出当前节点及子元素节点文本内容。
3. outerHTML：输出当前节点标签文本内容，连同子元素文本内容及子元素节点标签。

通过获取元素改变元素的css属性：

var x = document.getElementsByClassname("class")[0];

object.style.property="value";

x.style.class="red";

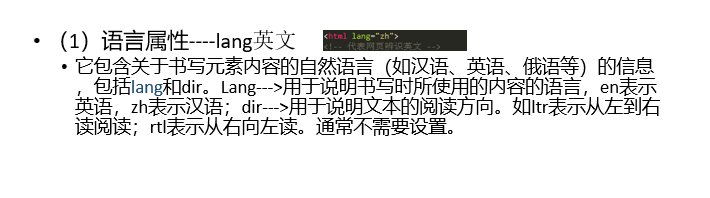
x.style.fontsize="15px";

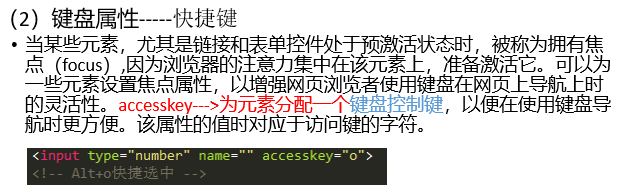
通过改变class名改变css属性：

（css属性分为样式属性和元素属性）

x.clsssName = "aaa";

//x.interHTML="通过innerhtml改变元素标签内内容"；//





数据类型-简单数据

简单数据分为5种

数字number、字符串string、布尔值boolean、空null、未定义undefined

字符串string------注意：加引号“”---可以单引号、双引号

字符串是字符的载体，而且必须被成对的引号包围。

如: 'hello world!' var data = 'hello world!';

typeof(data) 返回值是字符串 'string' ，即变量data的数据类型是 字符串

布尔值boolean

**true（真） 或 false（假）** 代表布尔值

**空null**

**若 typeof(data) 的返回值是字符串 'null' ，即变量data的数据类型是 null。即data为空**

未定义undefined

若 typeof(data) 的返 回值是字符串 'undefined' ，即变量data的数据类型是 undefined。即data不含有值。

数组语法：

[value1,value2,value3,.......valuen]

对象语法

{key1:value1,key2:value2,key3:value3,......keyn:valuen}

获取对象属性值1：myName=student.name；2: myName=student["name"]

数组arrey[a,b,c,d,e,f,g.........]

获取数组值：array[下标]。数组内数值类型随意，数值不限。

数据类型转换：①显示转换（强制类型转换）

1：字符串转数字。----

parseInt()取整，第一位不是数字返回NAN

parseFloat(), 第一位不是数字返回NaN，遇到第二个先个小数点或者遇到非数字时结束

Number()，包含非数字即返回NaN

2：转字符串。---

tostring(),null和undefined不可用。

String(), 可用于null及undefined。

3：转布尔值。

Boolean（），只要引号里有占位就会转换为true；引号里没有占位或是null及0时为false

②隐式类型转换（利用js弱变量类型转换）

1：使用减乘除，取余及比较运算符时，默认使用Number()将字符串转化为数字

2：使用加法运算会将数字看成字符串进行拼接。

前提：是数字与字符串混合的运算。

数值操作: 1，parseInt只截取整数，丢弃小数部分；

2，Math.ceil向上取整；

3，Math.floor向下取整；

4，Math.round四舍五入；

5，Math.abs绝对值；

6，Math.max Math.min返回两者中较大（小）值；

Math.max(a,b);

7,Math.random返回（0-1）之间的随机数；

isNaN用于判断是否是数字

null和undefined： 在if语句中都会被自动转化为false，相等运算符会报告

console.log(null==undefind);为true。

但在转化为数值时Number(null)=0;Number(undefined)=NaN。

JavaScript拥有动态数据类型，这意味着相同的变量可用作不同的类型：

Javascript中所有事物都是对象：字符串，数值，数组，函数。。。。。。。。

当函数被保存为一个对象的属性时，他可称为这个对象的方法

var x = {

info:"hello",

speak:function(){

alert("hello")

} }

访问对象方法：object.methodname(); console.log(x.speak());

==创建对象==

1：声明时添加；var x = {name:132,...........}

2: 声明后添加；var x = { }；x.name="!#@";console.log(x)

3: 通过Object原生构造函数：var x = new Object(); x.name="!@#";........

=======================运算符========================

按优先级

1：特殊运算符：. [ ] new ()

2: 算术运算符： ①自增自减：++ --

②乘，除，取余：

③ 加，减

3：比较运算符： ①> < >= <=

② == === !== !===

4：逻辑运算符：&& || !

5：赋值运算符：= \*= /= += -=

(三元(条件)运算符：condition？代码1：代码2；condition为true时执行代码1，false执行2)

特殊运算符：

new运算符：var today = new Date(); //创建一个当前系统日期时间对象。

星期：var day = today.getDay();

日期：var data = today.getDate()

年份：var a = today.getFullYear()；

月份：var a = today.getMonth()+1；（月份需＋1）

小时getHours；分钟getMinutes；豪秒getMilliseconds；

delete运算符：删除数组array的元素，或者对象object的属性。（注意：用delete方法删除数组元素时，**数组长度不变（empty空）**）

typeof运算符：一元运算符，主要功能：判断变量的数据类型的。如：typeof name 或 typeof(name)

Js流程控制：

1：顺序控制：即代码自上而下依次执行。

2：分支控制： Ⅰ：单路分支 if(){......};

Ⅱ：双路分支 if(){...}else{....};

三：多分支 if(){...}else if(){....}else......

Ⅳ：嵌套分支 if(){if().....}else{.....}

五：switch分支

switch(n){

case 1:

执行代码块1

break；

case 2:

执行代码块2

break；

default：与case1和case2不同时执行

}

3：流程控制： ①：for语句;for(循环初始值；循环条件；变量的变化){//需要执行的代码块}；符合条件时首先执行for循环中的代码块，再去对循环变量进行更新；

嵌套循环：for(){for(){.....}......};

for循环遍历数组：var array=["#@$","#%$R","%^#$"]; for(var i = 0);i <=array.length;i++){console.log(array[i]);}

②：for in循环,遍历对象属性:每次遍历时将对象的键依次赋给形式变量

\ var x = {key1:value,key2:value};for(i in x){document.write(x[i])}

三：while（条件）{//代码块}； 必须记得增加条件中所用变量的值， 不然会导致浏览器崩溃

④：do while：do{//需要执行的代码块} while(条件)

无论是否满足条件，先执行一次代码

##：break和continue：

break终止循环，使用break后其所在的循环体已结束；

continue用于跳跃循环中的迭代，如果出现了指定的条件，跳跃省略本 次循环然后继续循环中的下一次迭代，continue所在的循环体并没有结束；

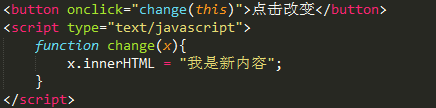
=========================================================

DOM事件：DOM使javascript有能力对HTML事件作出反应。对事件作出反应，指的是可以在事件发生时执行JS代码

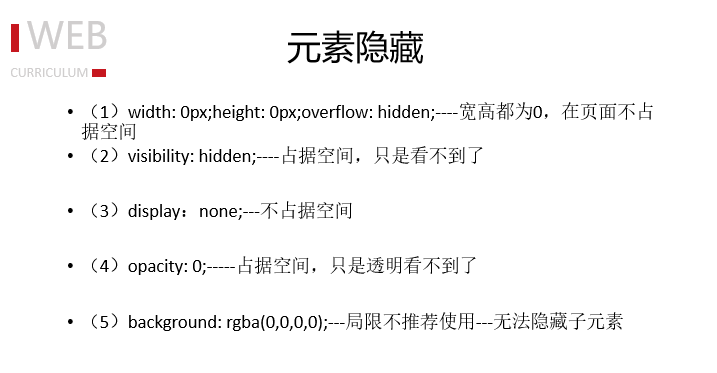
DOM事件模型

1. 内联模型/事件属性模型(和HTML 代码混合在一起，并没有与HTML 代码分开)





2，动态绑定/分配事件模型（DOM 允许使用 JavaScript 向 HTML 元素分配事件，首先获取元素，然后向元素节点分配事件。）



onclick鼠标单击；ondblclick鼠标双击；onmouseover鼠标移入；onmouseout鼠标移出

onmousedown 鼠标按下；onmouseup鼠标松开；onmousemove鼠标在区域内移动时触发

onmouseenter onmouseleave不支持事件冒泡

onkeydown 某个键盘按键被按下；

onkeypress 某个按键被按下并松开；

注意：在所有浏览器中 onkeypress 事件不是适用于所有按键(如： ALT, CTRL, SHIFT, ESC)。监听一个用户是否按下按键一般使用 onkeydown 事件,所有浏览器都支持 onkeydown 事件

onkeyup 某个按键被松开；

①onkeypress 事件在用户按下并放开任何字母数字键时发生。但是系统按钮（例如：箭头键、功能键）无法得到识别。

1. onkeydown 事件在用户按下任何键盘键（包括系统按钮）时发生。
2. 一个放开一个没有放开，onkeydown 先于 onkeypress 发生
3. 与 onkeydown 事件相关联的事件触发次序为
4. onkeydown--->②onkeypress--->③onkeyup

onblur：失去焦点时执行；onfocus：获得焦点时执行；

onchange：元素值改变时触发（适用于input，textarea select 等）

onsubmit‘：提交表单时触发；

oninput：用户输入时触发（类似于onchange，不同点在于onchange在元素失去焦点时执oninput是立即执行，且onchange可作用于<keygen><select>）

onpropertychange动态监听输入域的改变，类似oninput，但是他是IE的专有属性

onselect ：用户选取文本时触发（<input>和<textarea>）

onsearch：按下回车按键或者点击<input type=“search”>的搜索按钮时触发，兼容性较差

**粘贴板事件**

* （1）oncopy拷贝元素内容时触发
* （2）oncut剪切元素内容时触发
* （3）onpaste粘贴元素内容时触发

**窗口事件**

1、onload（load加载）页面加载完成后执行，<body onload="调用函数">

2、onresize（resize调整）窗口大小调整时执行，<body onresize="调用函数">

**函数**

函数（function）是由事件驱动的或者当它被调用时执行的可重复使用的代码块，函数就是将一段公共的代码进行封装，给它起个名字叫“函数”。函数在定义好之后多次调用。

常见的函数定义方式有多种。

1. 函数声明定义：function 开始，其后跟有 函数名、参数列表 和 函数体代码块。

function 函数名(参数, 参数, ...){ //代码块}

函数的调用：

1，<script>标签内调用；

2，HTML文档里调用（如通过点击按钮后调用定义好的函数

* 在调用函数时，可以向其传递值，这些值被称为参数。参数分为形参和实参，实参为实际应用的参数。可以调用任意多的参数，由逗号（,）分隔：参数可以有多个，但实参和形参在数量、顺序上必须严格一致，不然就会发生类型不匹配；发生函数调用时， 主调函数把实参的值传送给被调函数的形参从而实现主调函数向被调函数的数据传送

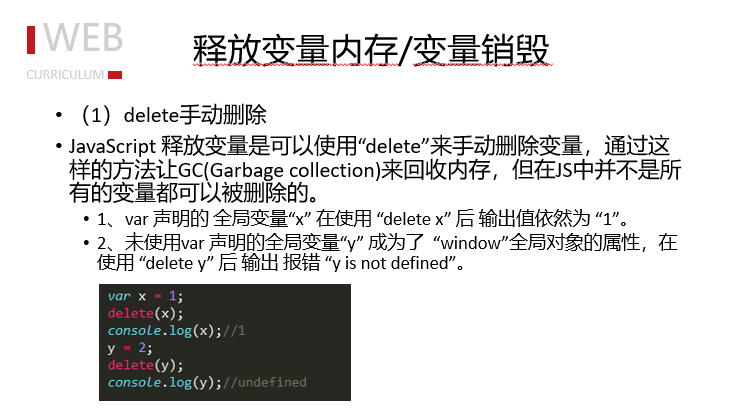
作用域的概念

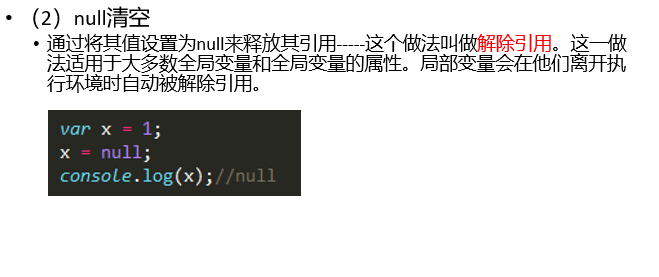
**作用域：任何程序设计语言都有作用域的概念，简单的说，作用域就是变量与函数的可访问范围，即作用域控制着变量与函数的可见性和生命周期。**

* Ⅰ，在JavaScript函数内部定义的变量（使用var）是局部变量，所以只能在函数内部访问它（该变量的作用域是局部的），即作用域范围只是函数内部。另外，形参也是局部变量。注意：只要函数运行完毕，本地变量就会被删除（变量生存期/生命长度）
* Ⅱ，全局JavaScript变量（2种）→可以在任意地方调用。

1. 在函数外声明的变量是全局变量，网页上的所有脚本和函数都能访问他。这个变量在任何函数中都有效，即作用域范围是当前文件的任何地方。即在最外层函数外面定义的变量拥有全局作用域
2. 向未声明的JavaScript变量分配值，末定义直接赋值的变量自动声明为拥有全局作用域。没有var进行定义的变量，写在函数中（注意：这个函数被调用一次之后，才能变成全局变量，如果不调用就解析不到）

释放变量内存/变量销毁[[1]](#footnote-1)





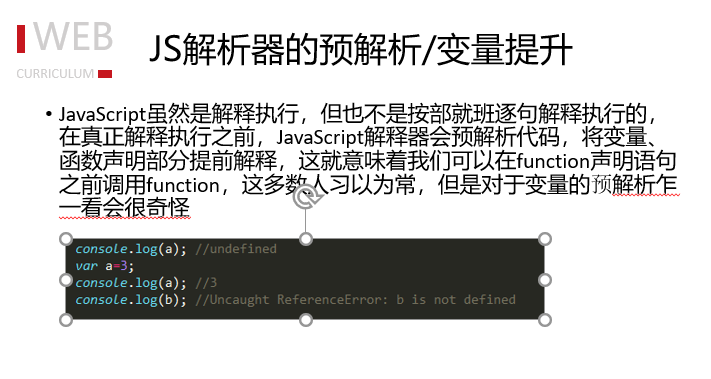
* 小结
  + js变量分为两种，一种是全局变量，一种是局部变量。全局变量，在js文件的任意地方都可以使用，它的生命周期就是js文件使用的周期。局部变量，它在所属的方法，或者说是它在自己属于的对象里面才存在，这个对象或者方法被解析过了，它就自动没有了。
  + 所以如果你要销毁变量，如果是全局变量，那么你需要覆盖它，如果是局部变量你只要等待它所在的方法结束，它就自动销毁了。将变量赋值为undefined 如果是个对象的属性，可以使用delete删除某对象属性或者方法

显示/隐示全局变量

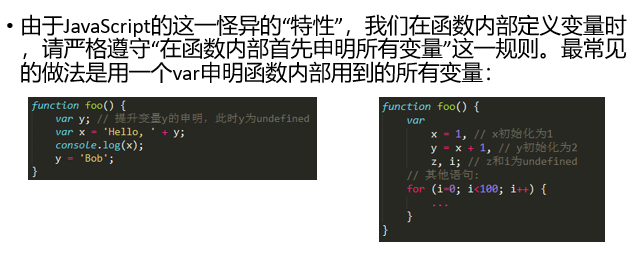
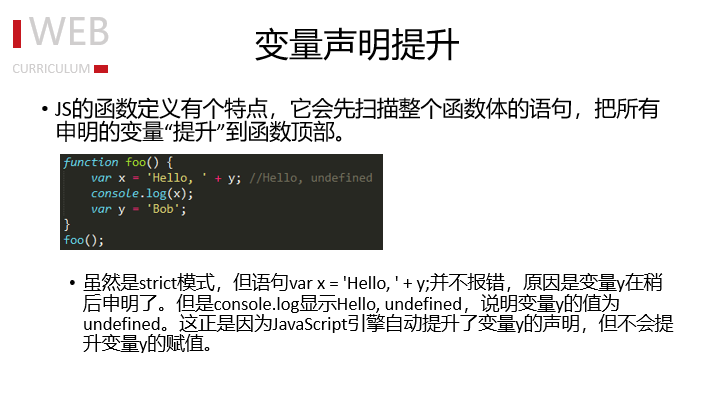
* 全局变量可以分为：隐式全局变量、显式全局变量。
* 隐式全局变量和明确定义的全局变量间有些小的差异，就是通过delete操作符让变量未定义的能力。
  + 1. 通过var创建的显式全局变量（任何函数之外的程序中创建）是不能被删除的。
  + 2. 无var创建的隐式全局变量（无视是否在函数中创建）是能被删除的。
* 这表明，在技术上，隐式全局变量并不是真正的全局变量，但它们是全局对象的属性。属性是可以通过delete操作符删除的，而显式变量是不能的。

变量的优先级

局部变量优先级高于全局变量：原理：函数执行时，会先从内层向外层查找，如果在内层中找到变量，查找就会停止。所以内层的作用域会优先于外层，局部优先于全局，这也是变量作用域链的原理。



* 上面代码在执行前var a=3; 的声明部分就已经得到预解析（但是不会执行赋值语句），所以第一次的时候会是undefined而不会报错，执行过赋值语句后会得到3，上段代码去掉最后一句和下面代码是一样的效果。



this对象概括

this关键字虽然会根据环境变化，但是它始终代表的是调用当前函数的那个对象。这就引出了JS中函数调用的问题。

函数调用方式

（1）方法调用模式

当函数被保存为一个对象的属性时，它就可称为这个对象的方法。当一个方法被调用时，this被绑定到这个对象上。如果调用表达式包含一个提取属性的动作（. 或 []），那么它被称为方法调用。

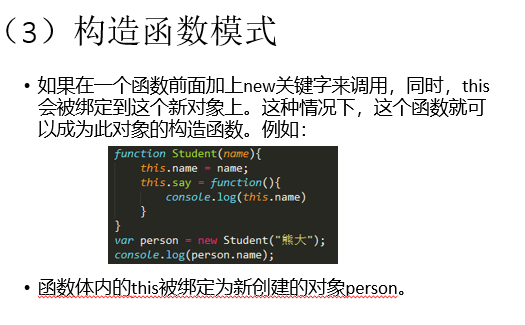
（2）函数调用模式

* 当一个函数并非一个对象的属性时，那么它就是被当做函数来调用的。在此种模式下，this被绑定为全局对象，在浏览器环境下就是window对象。

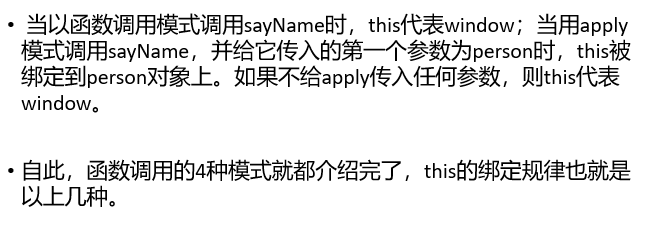
区别：

构造函数VS普通函数

* （1）开头字母
  + 
* （2）调用方式
  + 



 [[2]](#footnote-2)



**数组进阶操作**

常见定义方式

1. var x = new Array(3)(规定长度);

x[0]=1;x[1]=2;……

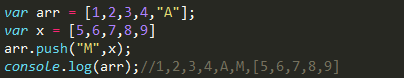
1. var x = new Array(length)
2. var x = [1,2,3,4,……]

打印数组时，如果某个元素没有赋值，则为“” ；

访问数组范围以外的元素时，不会出现越界异常，为undefined

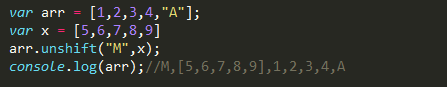
数组操作

**添加**

1. 通过字面量直接添加
2. push()方法--->向数组末尾添加，并返回新长度

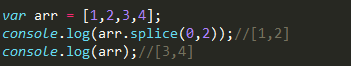
该方法也可以同时添加多个元素。

1. unshift()方法----->向数组开头添加，和push()类似，可以同时添加多个元素，并返回新长度



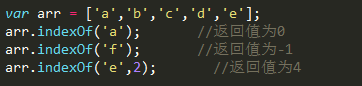
1. splice向指定位置插入元素

**删除**

1. pop()方法------->删除并返回数组的末尾元素，如果数组为空则返回undefine
2. shift()方法------>删除并返回数组的开头元素，与pop()相反
3. splice()方法----->可以删除任意数量元素并返回被删除元素。只需要指定2个参数：要删除的第一项的位置和要删除项的项数。

**修改**

1. 通过字面量直接修改
2. 通过splice()方法替换

**查找数组元素**

**正序查找**：indexOf()方法，也叫数组元素索引法，用于检测数组是否包含某个元素，并返回该元素索引，不存在的话返回-1

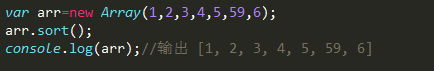
**语法**：arr.indexOf(searchvalue,fromindex)；应用：数组元素去重

**注意**：1.searchvalue => 必需，要检索的值；2.fromindex => 可选，元素开始检索的位置。如省略，则从数组的首个元素开始；3.indexOf()方法对大小写敏感！

1. **倒序查找：lastIndexOf()方法，从数组末尾开始查找**

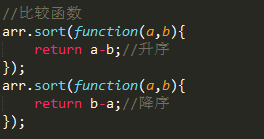
**lastIndexOf()方法虽然是从后往前搜索，但返回的位置是从前开始计数和计算的，所以结果和indexOf()方法返回的相同**

**排序**

sort() 方法用于对数组的元素进行排序。注意：sort()默认是转换字符串再排序， 所以直接按照数字大小排序不正确。

如果要得到自己想要的结果，不管是升序还是降序，就需要提供比较函数了。该函数比较两个值的大小，然后返回一个用于说明这两个值的相对顺序的数字。

* **比较函数应该具有两个参数 a 和 b，其返回值如下：1、若a<b，即 a - b<0，则返回一个小于零的值，数组将按照升序排列；2、若a=b，则返回 0；3、若a>b，即 a - b>0，则返回一个大于零的值，数组将按照降序排列；**

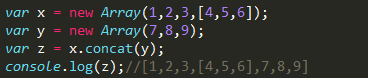
****

**倒序**

* **reverse()方法将数组中的元素倒置顺序**

**拼接**

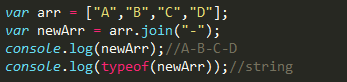
* **concat()方法用于拼接两个或多个数组。该方法不会修改数组而是返回一个新的数组,如果参数是数组，那么添加的是数组中的元素。注意：拼接数组中的多维数组是不会展开的。**

****

**截取/拷贝**

**slice(start,end);该方法不会修改数组而是返回一个新的数组。注意：start从0开始；包含start位置元素，不包含end位置元素；[[3]](#footnote-3)[[4]](#footnote-4)end可以省略，省略表示截取从start位置到数组结束**

* **注意：原数组没有改变。**

**转换**

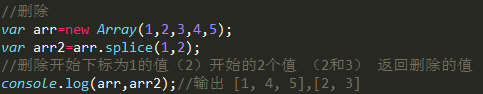
**1、字符串化；2、通过特定标记(可省略)拼接为字符串**

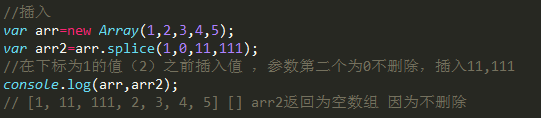
* + **1、toString()方法将数组转换成字符串类型**
  + **2、join(separator)方法通过分隔符拼接元素，间接转换成字符串。separator为指定要使用的分隔符，为可选参数，可以省略。（拓展：与join()方法对应的是string类的split方法[[5]](#footnote-5)，根据分割字符将字符串分割，之后将每一部分的字符串作为元素加入数组中，并将之返回。）**

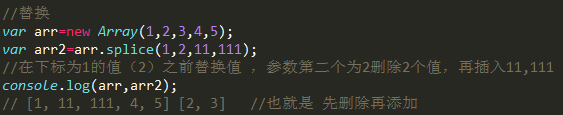
**剪接splice**

* **剪接splice（删除、插入、替换）**
* **splice()方法是一个很强的数组方法，它有多种用途，包含删除、插入、替换等，该方法会改变原始数组。**
* **语法：splice(index,howmany,item1,.....,itemX)**

1. **index => 必需。整数，规定添加/删除项目的位置，使；**
2. **howmany => 可省略。要删除的项目数量。如果设置为 0，则不会删除项目，如果省略，则会删除index位置及其后面的所有元素；**
3. **item1, ..., itemX => 可选。向数组添加的新项目。**

**1、删除**

**2、插入**

**3、替换**

**DOM遍历操作**

（1）childNodes返回一个数组，该数组由给定元素节点的子节点构成

语法： elementNode.childNodes

访问选定元素节点下的所有子节点的列表，返回的值可以看作是一个数组，具有length属性。如果选定的节点没有子节点，则该属性返回不包含节点的Nodelist。

（2）firstChild返回第一个子节点。语法：node.firstChild ，与elementNode.childNodes[0]是同样效果

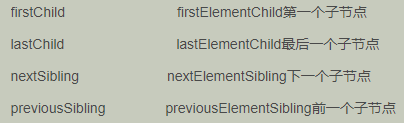
返回'childNodes'数组的第一个子节点，如果选定的节点没有子节点，返回null。

（3）lastChild返回最后一个子节点

（4）parentNode返回一个给定节点的父节点

（5）nextSibling返回给定节点的下一个子节点

语法：nodeObject.nextSibling ，无此节点，则返回null，返回某个节点之后紧跟的节点（处于同一树层级中）

****（6）previousSibling返回给定节点的上一个子节点

语法：nodeObject.previousSibling ，无此节点，返回null。

**加了一个过滤的效果，将换行节点过滤掉，只要元素节点**

**DOM常见操作方法**

**createElement(element)创建一个新的元素节点**

**createTextNode()创建一个包含着给定文本的新文本节点**

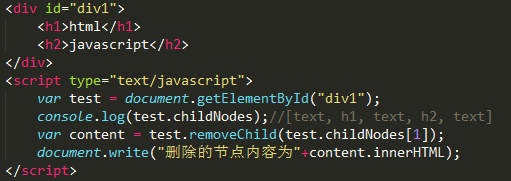
**appendChild()指定节点的最后一个子节点列表之后添加一个新的子节点**

**insertBefore()将一个给定节点插入到一个给定元素节点的给定子节点的前面**

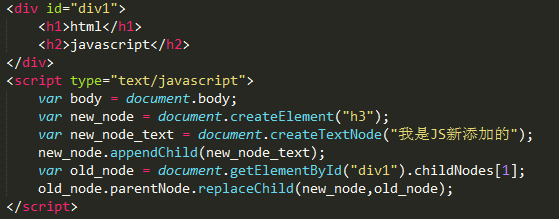
**removeChild()从一个给定元素中删除一个子节点**

**replaceChild()把一个给定父元素里的一个子节点替换为另一个节点**

**删除节点---removeChild()从子节点列表中删除指定节点**

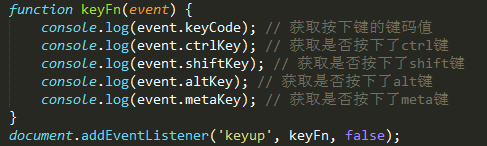


**替换节点---替换元素节点replaceChild()**



**改变 HTML 输出流**

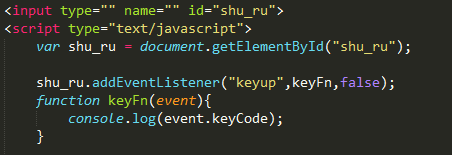
document.write() 可用于直接向 HTML 输出流写内容，这样会覆盖原有文档结构。所以不要在文档加载完成之后使用 document.write()，这会覆盖该文档。

**Event**

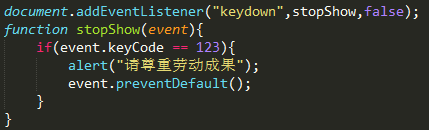
在用键盘触发事件时，主要的事件属性包含键盘的按键keyCode和是否按下特殊键，比如：keyCode指定了按下键的键码值，ctrlKey指定是否按下了ctrl键。

**Addeventlistener**

**查看键盘键码值----F12（123）**

****

**常见的一个应用就是禁止用户通过浏览器查看源代码（当用户在浏览器启动控制台时，禁止其查看源代码）**

****

**节点属性**

***nodeName：节点的名称、nodeValue：节点的值、nodeType：节点类型*。**

Event事件对象提供了一个属性叫target，可以返回事件的目标节点，我们称之为事件源，也就是说，target就可以表示为当前的事件操作的dom。当然，这个是有兼容性的，标准浏览器用event.target，IE浏览器用event.srcElement。

1. [↑](#footnote-ref-1)
2. [↑](#footnote-ref-2)
3. [↑](#footnote-ref-3)
4. [↑](#footnote-ref-4)
5. [↑](#footnote-ref-5)