# 2018.01.20

## 编程语言:

programming language计算机语言，人和计算机交流的单向语言。

为啥使用编程语言：

开发软件系统提高效率，比如： CRM（客户关系管理系统） ERP（企业计划计划管理系统） OA（在线无纸化办公系统）

## JavaScript

什么是JavaScript？是一门在客户端浏览器运行的脚本语言，也可以运行在服务器（Node js）

网页：内容/结构（html）/表现（css）/行为（js）

### JavaScript作用：

1. 表单数据验证
2. 动态HTML（DHTML）dom操作页面的结构、内容、样式
3. 交互操作：联动菜单....
4. 数据绑定：使用模板技术，把后台返回的数据呈现在页面上
5. 数据查询和返回，可以借助正则表达式实现
6. AJAX异步请求数据

### JavaScript开发工具：

webstrom----node.js

sublime

hbuilder----布局，js

visualCode---angular

ASP

### JavaScript的3种使用方式：

1. 行内脚本，直接写在标签上<h1 onclick=”alert(‘a li ga duo’)”>提交信息</h1>
2. 页内脚本，写在<script></script>之间
3. 外联JavaScript，先建一个外部\*.js文件，再引入<script src=”.js”></script>

### JavaScript的3种输出方式（掌握）

1. 浏览器对话框的方式（alert(“content”)）
2. 文档打印（输出在网页中）document.write(“content”)
3. 在调试面板的控制台输出 console.log(“content”)日志信息

### 注释的作用：

方便程序的阅读（团队协作）

单行注释：ctrl+/

多行注释：ctrl+shift+/

### 变量入门：

1. 数据存在：

内存：临时存储（退出程序/断电）

硬盘：永久存储

数据库：永久存储（调用数据更方便）

1. 什么是变量：

用于存储程序中变化的数据

1. 变量的作用

可以在内存中保存数据，通过变量名称多次使用该数据

1. 如何使用变量

语法：

var name;

var name=value;

var name1,name2,name3,....;

var name1=value1,name2,name3,....;（可以给一个或多个赋值）

var name1=name2=value;（定义相同值的两个变量）

### 变量的命名规则：

区分大小写

不能以数字开头，不能使用特殊符号

不能使用保留字和关键词

报错：捕获（uncaught syntaxError: unexpected token）

### 数据类型的作用：

1. 不同的类型分配不同的内存的空间，合理使用数据类型可以节约内存资源
2. 程序在执行中一般都是同种类型的数据进行运算

number（数字）：要进行数学运算的变量尽量使用数字类型

string（字符串）：电话号码，邮编使用字符串

Boolean（布尔）：安装条款（同意，不同意）

undefined（未定义）

null（空）

### 条件运算符（三目运算）

variable=condition?”ture\_value”: ”false\_value”;

### +运算符

var x=100;

var y=200;

var z=”300”;

alert(x+y+z)---->x+y=300(得到数字类型)+”300”(字符串)【将前面的300隐形转化成了字符串】=300300

alert(x+y+Number(z))-->600

alert(x\*y\*z)---->x\*y=20000(得到数字类型)\*”300”(字符串) 【将后面的300隐形转化成了数字】=6000000

### 数据转化

parseInt(var name)：把数字部分转化为整数

parseFloat(var name)：把数字部分转化为小数

Number(var name)：把纯数字字符串转化为数字

Boolean(var name)：转化成布尔值

String(var name)：转换为字符串

### 算术运算符

+

-

\*

/

%取模运算，求余数

++自身加一

--自身减一

### 比较运算符（返回的都是布尔值）

>

<

>=

<=

== 值 是否相等

!= 值 是否不相等

=== 全等 值和数据类型都相等

!== 不全等 值和数据类型都不相等

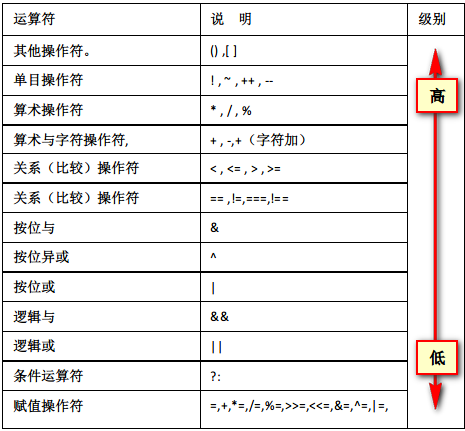
### 逻辑运算符

对布尔值进行运算

&&逻辑与

||逻辑或

!逻辑非（取反）



优先级

数学中的优先级：先乘除后加减，有（）先算（）内，从左到右依次运算

# 2018.01.21

## 作业评讲

### isNaN()

全局函数来判断一个值是否是NaN值; 不是数字true；是数字false

var x = “acsd”;

var y = 2000;

console.log(x, isNaN(x)); //true

console.log(y, isNaN(y)); //false

### ++i 和i++

var a=100, b=100, c=1;

var d1 = a++ + c;

var d1 = ++b + c;

console.log(“d1=a++ + c”, d1, a); // d1=a++ + c 101 101

console.log(“d2=++b + c”, d2, b); //d2=++b + c 102 101

## 课堂笔记

### 浏览器对话框：

1. 警告对话框window.alert(“msg”);
2. 输入对话框window.prompt(“提示信息msg”, “默认值default”)

prompt(“请输入你的成绩”, “80”); //但是光是这样的代码没有意义，要用变量装起来

var score=prompt(“请输入你的成绩”, “80”); //默认值可以为空字符串，视情况而定

点击确认就能获得用户输入的值，点击取消返回null

实际应用（根据输入的内容判断是否及格）：

var result = score >=60?”congratulation”:”loser”（这是不考虑输入不正确的情况）

alert(score+” : ”+ result);

1. 选择对话框window.confirm(“提示选择信息”)，确认返回true，取消返回false

var isAgree=confrim(“你愿意娶tamaki吗？”)

alert(isAgree);

window代表窗口对象，属于顶层对象，一般可以省略不写

### 流程控制的类型

1. 逐行执行：由前至后依次执行
2. 分支流程结构：会根据条件判断有选择性的执行相关任务
3. 循环流程结构：代码根据条件，重复执行

#### 条件语句

##### 单分支if语句

if(condition){

condition=true excute the code block here

}

##### 双分支

if(condition){

condition=true excute the code block here

}else{

condition=false excute the code block here

}

三目运算就是简化的if...else双分支语言

##### 多分支

3.1 else if

if(condition1){

condition1=true excute the code block here

}else if(condition2){

condition2=true excute the code block here

}else if(condition...){

condition...=true excute the code block here

}else{

condition cannot match all the situations above then excute the code block here

}

3.2 switch（状态分支语言）

switch(condition){

case value1:

code block1;

break;

case value2:

code block2;

break;

....

case valueN:

code blockN;

break;

default:

condition cannot match all the situations above then excute the code block here

break;(this break is optional)

}

var week=Number(prompt(“请输入0-6的数字：（0代表周天）”, “”));

alert(typeof week); //没有加上Number的数据类型是string

switch(week){

case 0:

alert(“周末”);

break;

....

}

#### if else语句和switch语句是区别：

相同：都是多分支语句

不同：if中条件可以是一个范围，switch条件只能是单个值

状态太多switch的状态值太多写起来很麻烦

## 循环语句

### for循环

for(初始值;条件表达式;改变值的表达式){循环体}

循环三要素：初始值 循环条件 每循环一次一定要改变变量的值

### while循环

var 变量=初始值

while(){

循环体

改变变量的值

}

### do-while循环

var 变量=初始值

do{

循环体

改变变量的值

}

while(condition)

不管条件是否满足，至少循环一次

### 嵌套循环

循环中的循环

for(var x=1; x<101; x++){

for(var y=1; y<101; y++){

}

}

嵌套的作用：目前阶段 多行多列的输入（表格）

列在行内

分开写行的循环和列的循环，再合并

### 终止循环语句

#### break

终止当前循环，后面的循环不再执行（找到嫌疑犯）

#### continue

终止单次循环，后面的循环还会继续执行（有不在场证明的排除）

# 2018.1.23

## 课堂笔记

### 什么是函数？

1. 代码的封装，被封装的代码具有某项功能。例如Number()函数的功能就是将纯数字字符串类型转为数字类型的函数。

### 函数的作用

提高代码的重用性，可维护性

### 如何定义函数

#### 定义的语法：

优先级 function高于var(作用域的优先级高于变量),

function name() {

//函数体的内容：封装的代码

}

or

相当于把函数名提前作为一个变量名，然后把匿名函数赋值给变量

var name = function() {

//函数体的内容：封装的代码

}

#### 调用的语法：

函数名name()

注意：函数只有调用后才发挥作用，实现函数功能

#### 带参数的函数：

当数量或者变量是未知的，此时需要参数。如：求任意两个数的和

参数的说明：

形参：在定义函数时传入的变量参数

实参：在调用函数时传入的变量参数

例子：

function add(num1,num2,num3) {

document.write(“任意三个数的和：”+num1+num2+num3);

}

调用方法一：

add(10,53,56);

调用方法二：

var num1 = parseInt(prompt(“请输入第一个数：”));

var num2 = parseInt(prompt(“请输入第二个数：”));

var num3 = parseInt(prompt(“请输入第三个数：”));

注意：这里的num123依然是实参

#### 函数返回值

函数功能代码执行的结果参与二次运算或者被其他程序调用，所以将结果返回，使用return语句。

调用的语法：

var result=name(param1,param2...)

定义一个变量来接收函数的返回值

#### 定义函数的三部曲

1. 根据功能/需求来确定函数名，命名尽量有意义
2. 是否有未知数据参与运算，决定是否使用参数
3. 函数内部输出还是外部使用结果，决定是否需要返回值

字符串的拼接，不能用+=

### 变量的作用域

变量在哪个范围内有效

作用域分类:

全局变量：在函数外部定义

局部变量：在一个函数内部生效

特殊情况：

在函数内部定义变量时，没有var关键词，也会是全局变量

### 系统函数

JavaScript语言在指定初期就定义好的函数（内置函数），我们不需要定义可以直接使用。

Stiring Number Boolean parseInt parseFloat isNaN eval(字符串):将字符串作为表达式运行。

eval通常用在JavaScript模板

### 递归函数

一个函数在函数内部自己调用自己。

作业：大问题拆成小问题，小问题与大问题的解决思路一致

应用：树状结构读取文件目录

注意： 要有递归点（何时调自己）

要有递归出口（何时停止调自己）

function recurrence(n) {

document.write(n);

n++;

if(n<10) {

recurrence(n);

}else{

console.log(“递归出口，递归调用结束”)

}

}

实例：递归算法打印1-n的累加，n表示正整数

# 2018.01.24

## 课堂笔记

### 数组

#### 数组的概念：

一组数据的有序排列的集合collection。一种复合数据类型，也叫引用数据类型，可以把多个数据和不同类型的数据存储在一个数组中来使用。

#### 数组的专业术语：

数组元素：组成数组的每一个数据。

数组索引：index 也叫下标，对元素的编号，从0开始，最后一个索引length-1。

数组长度：length组成数组的数据的总个数。

访问数组元素：arrayName[index];

### 一维数组

#### 定义一维数组：

var array1=[e1,e2,e3....];

var array2=[]; //空

var array3=new Array(e1,e2,e3....);

var array4=new Array(); //空

var array5=new Array(5); //长度为5的空数组

#### 增加修改数组元素

arrayName[index]=value;

索yin存在就是修改，不存在就是新增

#### 删除数组元素

delete arrayName[index];

清空数组：

arr =[];

arr.length =0;

#### 数组的遍历

遍历：traversal，依次对数组中的每一个元素进行单次访问

遍历的语法：

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

arrayName[i];

}

for(var i in arrayName) { for(变量 in 对象)

arrayName[i]; 执行代码

}

### 二维数组

二维数组本质上是以数组作为数组元素的数组。【把一维数组作为另一个数组的元素】

var arrayName=[

[e1,e2,e3],

[e4,e5,e6],

[e7,e8,e9],

[e10,e11,e12]

];

二维数组的作用：

多行多列的数据存储，如：所有学生的科目成绩

访问二维数组

arrayName[indexOut][indexIn]

增加修改删除清空二维数组，同一维数组。

遍历二位数组。遍历行（外）遍历列（内）

嵌套循环：

for(var x in 数组名) {

console.log(“行的元素：”,数组名[x]);

for(var y in数组名[x]) {

console.log(“列的元素：”,数组名[x][y]);

}

}

### 数组自身常用的方法

#### Array对象属性：

constructor length prototype(原型链)

#### Array对象方法：

concat() 连接两个或以上的数组，返回一个新的数组 array1.concat(array2);

join(separator) separator（optional, default,）将数组的所有元素放入字符串，返回一个字符串

array=[1,2,3] array.join(“~”); 1~2~3

pop() 删除并返回数组的最后一个元素 array.pop();

shift() ..............................第一个.......... array.shift();

splice(index.howmany,item1...,itemX) 添加或删除项目，然后返回被删除的项目。

index 必选。开始位置。

howmany 必选。删除的数量（后面有item的话就意味着更改），可以为0（后面有item的的话意味着添加）

item1-x 可选

push() 向数组的末尾添加一个或者多个元素，并返回新的长度 array.push(newEle);

reverse() 颠倒数组顺序 array.reverse();

sort(l) 排序（默认ASCII由小至大，数字依然以第一位的ASCII码来排序）

sort(sortby); 需要以其他的方式来排序，需要提供比较函数

function sortASC(a,b){return a-b;}

function sortDESC(a,b){return b-a;}

array.sort(sortASC); //升序

array.sort(sortDESC); //降序

# 2018.01.26

## 复习：

### 对话框：

alert(“警告”);

prompt(“输入”);

confirm(“选择”);

### 函数

function fName() {} function fName(x,y,z) {alert....} function fName() {return...}

var fName=function() {}匿名函数

## 对象

对象就是一组属性与方法的集合。

对象是一种复合数据类型，或者也称为引用类型。

### 对象的创建

#### 语法1：new

var objectName =new Object();

objectName.propertyName = propertyValue;

objectName.functionName = function(){

//content

}

#### 语法2：字面量

var objectName = {

propertyName1 : propertyValue1,

propertyName2 : propertyValue2,

functionName : function(){

//content

}

};

### 对象的分类

#### 内置对象：

JavaScript自带的对象，无需定义，直接使用。String Number Math Date

#### 自定义对象：

使用new和字面量，创建出来的对象

### 对象中的this

this指向的是对象本身，谁调用这个方法，this就指向它。

this 指向对象的好处在于可以更加方便地访问对象的内部成员。

## String对象

var str1 =”abc”; typeof str1--> string str1-->abc

var str2 = new String(“abc”); typeof str2--> object str2-->String{“abcd”}

### 字符串对象的属性：

length 字符串的长度

stringObject.length

应用场景：

表单验证：要求用户名的长度6-12位

### 字符串对象的常用方法：

#### indexOf()

子字符串在父字符串首次出现的位置。没有检索到，返回-1

stringObject.indexOf(searchvalue,[fromindex]);

#### lastindexOf()

最后出现的位置。【判断文件格式】

#### replace()

用新字符串替换旧字符串，也可以用正则表达式

stringObject.replace(reg,replacement)【留言的过滤功能】

#### split()

把字符串分隔成字符串数组。

stringObject.split(seperator,[howmany])

#### toLowerCase() toUpperCase()

toLowerCase转小写 toUpperCase转大写

#### substr subString

substr() 在字符串中提取

stringObject.substr(start,[length]) 从哪里开始取，可选取多长，不写的话默认取到结尾

substring() 提取字符串

stringObject.substring(start,[stop]) 从哪里开始取，哪里结束（可选），默认到结尾

应用场景：新闻标题过长折行，影响版面美观，对标题进行截取。

## Number对象

### Number的属性

### Number的方法

toFixed(小数的位数) 把Number四舍五入为指定的小数位数的数字

应用场景：购物车计算，会涉及到小数位数

## Math对象

数学运算的方法定义在Math对象中。

math不需要new，直接调用

### Math的属性

E PI

### Math的方法

abs(x) 绝对值

ceil(x) 上舍入 只要有小数，整数加一

floor(x) 下舍入 有小数，只保留整数

round(x) 四舍五入

max(x,y) min(x,y)

random() 返回0-1之间的随机数

指定范围的随机数公式：Math.floor(Math.random()\*(max-min+1)+min)

## Date对象

应用场景：订

机票\酒店\车票....

var dateObject = new Date(); //本地系统当地时间

var dateObject = new Date(“字符串的日期”); //指定一个日期

var dateObject = new Date(year, month, date[,hours,minutes, seconds,ms]);

### Date的方法

get 获取

set 设置

to 转换

getDate() 返回一个月中的某一天（1-31）

getDay() 返回一周中的某一天（0-6,0是周天）

getMonth() 返回月份0-11 -----》自己使用的时候需要+1

getFullYear() 以四位数返回年份

getHours() 小时0-23

getMinutes() 分钟0-59

getSeconds() 秒0-59

getMillisecond() 毫秒0-999

set类似

toString() 转换字符串

toLocalString() 根据本地时间格式，转换为字符

toTimeString() 将时间部分转换成字符

toDateString() 将日期部分转换成字符

# 2018.01.27

## callback回调函数

简称：cb，将一个函数作为参数传给另一个参数。

### forEach() 遍历数组内容。

foreach(function(value[, index, array]) {

//....回调函数的内容

});

参数说明：

value 数组的元素值，顺序一定一致

index 数组索引

array 数组本身

### map()

映射（一一对应）

map遍历数组，的用法跟foreach一样，不同之处是map会产生新数组

var newArr = arrName.map(functionvalue(value[, index, array]) {

//....回调函数的内容

return

});

### filter()

过滤并得到新数组，语法和map类似

var newArr = arrName.filter(functionvalue(value[, index, array]) {

//....回调函数的内容

return

});

## JavaScript的组成部分

### JavaScript=ECMAcript + DOM + BOM

ECMAScript：核心语法

BOM：浏览器对象模型，操作浏览器 browser object model

DOM：文档对象模型，操作html，css，xml文档的结构和内容 document object model

### BOM：

window（顶层对象）-------document文档对象------DOM模型

-----history访问历史步骤

-----location地址栏

-----navigator浏览器内核及相关信息

-----screen 客户端屏幕对象

-----frames[] 框架集

#### window对象的属性方法

顶层对象，浏览器窗口对象

属性：

length：设置或返回窗口中的框架数量

parent：返回父窗口

self：返回自身窗口

top：返回顶层窗口

方法：

alert() 警告对话框

close() 关闭浏览器窗口 onclick=”window.close()”

confirm() 选择对话框

open() 打开新的浏览器窗口 onclick=”window.open(url)”

print() 打印窗口内容 onclick=”window.print()”

prompt() 输入对话框

#### 定时器

setInterval(cb,ms) 设置周期性的定时器

clearInterval(timeID) 取消由setInterval设置的定时

setTimeout(cb,ms) 设置单次定时器

clearTimeout(timeID) 取消由setTimeout()设置的定时

参数说明：

timeID 定时器的id名称

cb 定时器的回调函数，定时器里面要执行的代码块

ms 定时器的间隔时间，多少毫秒后执行回调

例子：

等待5秒后，显示活动开始

setTimeout(function(){

console.log(“%c活动开始”,”color:red;font-size:50px;”);

},5000);

倒计时10秒，公布中奖名单

var num = 10;

var timeID = setInterval(function() {

console.log(num);

num--;

if (num==0) {

console.log(“%c中奖人是：你”,”color:red;font-size:50px”);

clear.setInterval(timeID);

}

}, 1000);

#### screen对象

screen屏幕信息

属性：

availWidth 可用屏幕宽度

availHeight 可用屏幕高度

应用场景: 用户显示器的分辨率情况统计，从而决定网页的主流设计宽度

#### navigator对象

navigator浏览器的各种信息

属性：

userAgent：用户代理头，记录浏览器核心和版本信息

cookieEnabled：true\false 是否支持cookie

应用场景：统计用户使用的浏览器的占有率情况。

两种方式都可以调用属性

var browser = navigator.ueserAgent;

var browser = navigator[“userAgent”];

例子：

if(browser.indexOf(“WebKit”)>0) {

document.write(“这是Chrome谷歌浏览器”);

}else if(browser.indexOf(“WebKit”)<0 && browser.indexOf(“Geoko”)>0 && browser.indexOf(“Trident”)<0) {

document.write(“这是火狐浏览器”);

}else if(browser.indexOf(“Trident”)>0) {

document.write(“这是IE微软浏览器”);

}

#### location对象

属性：

href设置--跳转网页或者获取url地址。

方法：

reload重载网页，等价于F5

#### history对象

历史的步骤数，指前进和后退，不是右键历史记录

属性：

length，步骤的（url）个数

方法:

back()，后退功能

forward()，前进功能

go(num)，跳转（num正数往前，负数往后）

### DOM:

document object model文档对象模型。可以动态访问和更新文档的内容，结构和样式

#### DOM的分类

核心DOM：公共接口，处理树结构。

XML DOM：操作XML DOM，操作XML文档

HTML DOM：操作HTML文档

HTML DOM事件模型：事件响应，让网页具有交互能力

CSS DOM：操作css样式

#### 节点树

DOM节点的大小关系：document(整个.html文档)>html>body>div(div,ul,p...)

获取body的方式：document.body;

document.getElementsByTagName(“body”)[0];

document.getElementById(“myBody”);

document.documentElement 表示HTML根元素，是整个窗口

document.body 表示body网页的主体

例子：

获取页面的高度

var h1 = document.documentElement.clientHeight

var h2 = document.body.clientHeight

#### 节点的类型

一个.html文档就是文档节点document

每个标签就是一个元素节点element

标签的属性就是属性节点attribute

标签中的文字就是文本节点（包括回车和空格）text

一切都是节点node

XML：extensible Markup language

提供跨平台的轻量级数据交换。

构成：头文件<?xml version=”1.0” encoding=”ISO-8859-1”?> +自定义标签+必须有根元素

#### HTML DOM中访问元素的方法（6种）

HTML DOM的作用，改变文档结构，修改文档内容，操作css样式

访问元素的6种方法：

前三种不存在兼容性问题

1. document.getElementByID(“元素的id”) 根据标签的唯一ID来查找元素
2. document.getElementsByTagName(“标签名称”) 得到标签名称的集合（一个元素也是集合）
3. document.getElementsByName(“标签的name属性名称”)

以下三种慎用

1. document.getElementsByClassName(“类的名称”)
2. document.querySelector(“选择器的第一个”) document.querySelectorAll(“选择器的所有”)
3. 快速访问，images links forms所有表单的集合

document.images[1].src=”新的路径”;

#### 常用的css属性和表单属性操作

1. checked 单选或多选的选中状态
2. selected 下拉菜单的选中
3. className 操作类样式
4. style 操作行内样式
5. readonly 设置表单的只读方式 ---》readOnly 应用：不允许用户修改
6. disable 设置是否被禁用---》true 应用：验证码，倒计时之前禁用

# 2018.1.29

## 课堂笔记

### 表单对象

属性：

action 处理表单的页面的url地址。可以通过程序修改表单的提交地址。

方法：

submit() 通过结合setTimeout让程序在指定的时间自动提交表单。

value 设置或者返回文本域的value属性值。

### 表单元素对象input

属性：

readOnly 只读

disabled 禁用

方法：

focus() 设置文本域焦点。第一次加载设置默认焦点，表单验证数据出错设置默认焦点。

### HTML DOM Node

复习：访问节点的6种方式

document.getElementById();

document.getElementsByTagName();

document.getElementsByName();

document.getElementsByClassName();

document.querySelector() document.querySelectorAll();

images links forms

#### 节点信息

|  |  |  |  |
| --- | --- | --- | --- |
|  | nodeName | nodeValue | nodeType |
| element(元素) | 标签名 | null | 1 |
| Attribute（属性） | 属性名 | 属性的值 | 2 |
| Text（文本） | #text | 文本（包含回车和空格） | 3 |

#### 遍历节点

parentNode 父节点

firstChild 列表中的第一个节点

lastChild 列表中的最后一个节点

firstElementChild 第一个元素节点

lastElementChild 最后一个元素节点

childNodes 所有子节点列表（集合）

previousSibling 上一个兄弟节点

nextSibling 下一个兄弟节点

previousElementSibling 上一个兄弟元素节点

nextElementSibling 下一个兄弟元素节点

#### 删除节点

方法：

1. parent.removeChild(child[index]) 找到父节点，删除指定索引的子节点
2. node.remove() 找到要删除的节点，直接调用remove()

批量删除：

1. parent.innerHTML =””; 父节点中的内容为空
2. parent.hasChildNodes() 父节点是否有子节点，有true，无false

while(parent.hasChildNodes()){

parent.removeChild(parent.firstChild);

}

#### 添加节点

node节点的操作--添加节点

...............................................内存中..............................................

document.createElement(“标签名”) 创建元素节点

document.createTextNode(“文本的内容”) 创建文本节点

elementNode.setAttribute(属性名称，属性值) 设置属性

elementNode.getAttribute(属性名称) 获取属性

...............................................页面中..............................................

parent.appendChild(childNode) 在父节点中追加子节点（末尾）

parent.insertBefore(newNode, refNode) 在参照节点之前插入节点（任一位置）

#### 替换节点（修改）

parent.replaceChild(newNode,oldNode) 将父节点中的旧节点，替换为新节点

#### 克隆节点（复制）

target.cloneNode(deep)

deep---Boolean----true克隆需要的属性和子节点(后代)

false（只克隆父节点）不需要克隆属性和子节点----默认

### InnerHTML innerText

修改元素内容

node.innerHTML = “html content”

node.innerText = “text” 纯文本

获取元素内容

node.innerHTML

node.innerText

### 事件 event

用户通过鼠标\键盘\语音设备等方式，与计算机产生交互。实现用户与计算机的交互的方法，人机互动的桥梁。

#### 鼠标事件：

onclick 单击鼠标

onmousedown 按下鼠标

onmousemove 移动鼠标

onmouseout 鼠标从某元素移开

onmouseover 鼠标移动到元素之上

onmouseup 松开鼠标

#### 键盘事件

onkeydown 键盘被按下

onkeypress 键盘被按住

onkeyup 键盘被松开 keyNode.onkeyup=function(){

console.log(even,”evencode能得到按的键”,evencode)

}

#### 表单事件

onfocus 获得表单元素的焦点

onblur 表单元素失去焦点

onsubmit 提交表单 formNode.onsubmit=function(){

alert(“可以阻止表单提交，只要return false”)

return false;

}

#### 窗口事件

onload 页面加载完毕 window.onload=function(){...} script此时可以放到最后

onresize 改变窗口尺寸

onscroll 正在滚动浏览器的滚动条

# 2018.01.30

## 课程笔记

### 事件句柄 = 事件处理函数

事件要发生时要发生时要进行的操作，即是说，事件执行一个函数。

每一个事件都对应一个事件句柄，在程序执行时，相应的函数或语句指定给事件句柄。

An event handler is a callback routine that operates asynchronously and handles inputs received into a program (events). In this context, an event is some meaningful element of application information from an underlying development framework, usually from a graphical user interface (GUI) toolkit or some kind of input routine. On the GUI side, events include key strokes, mouse activity, action selections, or timer expirations. On the input side, events include opening or closing files and data streams, reading data and so forth.

事件处理程序是一个回调例程，异步运行并处理接收到程序（事件）的输入。 在这种情况下，事件是来自底层开发框架的应用程序信息的一些有意义的元素，通常来自图形用户界面（GUI）工具箱或某种输入例程。 在GUI方面，事件包括击键，鼠标活动，动作选择或计时器到期。 在输入端，事件包括打开或关闭文件和数据流，读取数据等等。

### 事件注册的三种方式（important）

#### 直接在元素上添加事件

<img src=”...” onclick=”alert(‘click’);” />

<img src=”...” onmouseover=”this.src=’...’;” onclick=”alert(‘click’);” />

优点：简单直接 缺点：入门，不利于后期代码维护

#### 在元素标签的DOM对象上添加事件句柄函数，并赋值一个句柄函数event handler（recommend）

两种方式：

1. 先定义function functionName () {...}

DOM元素对象.事件名称=functionName();

1. 使用匿名函数

DOM元素对象.事件名称=function() {...}

例子：

鼠标移入移除不同的背景颜色

var div = document.getElementById(“divId”);

function mouseOver(){div.className=”divHover”;}

function mouseOut(){div.className=””;)

div.onmouseover=mouseOver;

div.onmouseout=mouseOut;

#### 在DOM元素对象上使用监听（侦听）

低版本的IE（了解）

attachEvent 添加事件

detachEvent 移除事件

W3C标准做法（掌握）

添加语法：

target.addEventListener(type, listener, useCapture);

target: DOM元素，往谁上添加事件

type：事件名称，不含on，是保留click

listener：事件执行函数

useCapture:是否冒泡，默认false不冒泡，true要冒泡

例子：

var ul= document.getElementById(“ulId”);

ul.addEventListener(“click”, function(){

alert(“....”);

},false);

移除事件：

target.removeEventListener(even, function, useCapture);

ul.removeEventListener(“click”, functionName,false);

#### 事件处理函数返回值

事件发生时，浏览器会自动执行默认动作。比如: 超链接，表单提交....

事件句柄不返回值，或者返回true，就执行默认动作。返回false，阻止动作执行。

### 事件的捕捉和事件冒泡

事件冒泡：

当事件发生时事件通过元素的关系进行传播（由里到外或者由外到里）

为什么要传播？因为事件源本身（可能）并没有处理事件的能力，即处理事件的函数（方法）并未绑定在该事件源上。

离事件源开始冒泡。

useCapture:默认false，是我们想要的效果（由里到外）

### even事件对象

event对象：就是事件对象，代表一个网页上的事件发生时的相关信息。

比如：事件发生的来源，鼠标的位置，键盘触发了哪个键

作用：可以使用同一个方法（句柄函数），但获取到不同的事件源对象的相关信息。

比如：电子积分牌

event对象的属性和方法：

属性：

event.clientX 鼠标横坐标的位置

event.clientY 鼠标纵坐标的位置

target 事件源，哪个元素触发了事件

type 事件的类型名称

方法：

preventDefault() 阻止浏览器的默认行为

stopPropagation() 阻止冒泡

function functionName(e) {

var e = e || window.e

}

### 上下文菜单事件（鼠标右键菜单事件）

oncontextmenu

//在浏览器窗口注册上下文事件

方法1：

document.documentElement.oncontextmenu=function(){

alert(“...”);

}

方法2：  
 window.oncontextmenu=function(){

alert(“...”);

}

使用oncontentmenu的时候，浏览器的右键菜单也会出现。

解决办法：

window.oncontextmenu=function(e){

alert(“...”);

var e = e || window.event;

e.preventDefault;

// return false; 更简单的方式

}

定位菜单（top left）

隐藏右键菜单

e.clientX 赋值给top 和right

e.clientY

### 正则表达式

正则表达式 regular expression 也称规则表达式

是一种匹配规则的模式。在JavaScript中，是RegExp正则对象。

正则表达式的作用：

通常用来检索，替换符合规则的文本。比如：表单验证，字符过滤，爬虫程序...

定义正则表达式：

1. 字面量定义方式

var re = /规则定义/;

1. 对象的定义方式

var re =new RegExp(规则定义)

正则表达式常用的方法：

1. test 检索目标字符串是否符合匹配规则，匹配true，否则false

re.test(“目标字符”)

1. replace替换与正则表达式匹配的子字符串

string.replace(re,“替换后的字符串”)

正则表达式的修饰符

修饰符 描述

i 忽略大小写

g 执行全局匹配

m 执行多次

修饰符放置的位置 var re=/规则的定义/igm

常用的正则表达式：

# 2018.02.01

## 复习

### 事件注册与捕获

#### 事件注册的三种形式

1. 直接在元素上添加句柄函数

<button type=”button” onclick=”alert(‘点啥？’)”>按钮</button>

1. 在元素标签的DOM对象上添加事件句柄函数属性

<button type=”button” id=”btn”>登陆</button>

<script type=”text/javascript”>

var btn = document.getElementById(“btn”);

btn.onclick=function(){

....

}

</script>

<script type=”text/javascript”>

var btn = document.getElementById(“btn”);

function login(){

....

}

btn.onclick=login;

</script>

1. W3C标准添加事件

target.addEventListener(type,listener,useCapture);

target:文档节点，document，window或者XMLHTTPRequest

type:事件名称，不要“on”

listener：函数

useCapture：是否冒泡，默认false

var btn = document.getElementById(“btn”);

btn.addEventListener(“click”,function(){

alert....

},false);

### Event对象

event对象，也就是“事件对象”：代表一个网页上的事件发生时的相关信息。

事件源：事件发生在谁身上。

兼容性问题：Firefox的事件必须将event对象作为参数传入函数中。

注册事件：

需要html的对象: document.documentElement--->标签

需要body的节点：document.body--->节点

//获取对象：

var mx = document.documentElement.clientX;

var my = document.documentElement.clientY;

document.documentElement.onmousemove=function(e){

//兼容性

[var]e =e || window.event

var mx = e.clientX;

var my = e.clientY;

//动态的加到指定的div中

//找到先

var divBox = document.getElementById(“divBox”);

//将动态的值传入到box中

divBox.innerHTML=mx+”, ”+my;

}

### 正则表达式

正则表达式：以一个用一个简单方式描述或者匹配一系列符合某个句法规则的字符串。

通常用来检索，验证用户输入信息是否满足规则。

1. 检查字符串是否合法
2. 提取字符串（后缀，信息中的数字）
3. 替换字符串（非法字符替换，隐私\*）

正则表达isRegExp对象

方法：1. 使用new构造方法， var r = new RegExp(“test”);

2. 使用字面量， var r = /test/;

例子：

var text = document.getElementById(“text”);

var textRE = /^.{3,20}$/;

textRE.test(text.value)--->返回布尔值

if(!textRE.test(text.value)){

alet(...);

text.focus();

return false;

}

正则表达式

性别判定，getElementsByName()-->数组

! gender[0].checked && gender[1].checked

区域判定：option不为空判定

所有的option都，要给value值，其中“请选择”的value是-1，其它为缩写

city.value==-1{alert}

tabindex 属性规定元素的 tab 键控制次序（当 tab 键用于导航时）。

# 2018.02.02

## 返回顶部：return top

返回顶部的功能分析：

1. 滚动条达到指定高度时，显示返回顶部的按钮。
2. 点击 返回顶部按钮，实现返回到顶部。

步骤：

1. 先网页加载时实现某事件window.onload
2. 再在window.onload下实现window.onscroll事件
3. window.onscroll事件下，在要获取滚动条的高度，可能遇到兼容性的问题（是选取使用通过HTML节点拿到高度document.documentElement.scrollTop，~~还是通过body拿到高度document.body.scrollTop）~~

注意：有些值在取的时候 是不带单位的。比如这里的高度。

1. 找到对象，判断3中取到的高度，大于某值display=”block”, else{display=”none”}
2. 单独实现点击注册事件xx.onclick=function(){

//为了实现循环的减去自身高度，使用定时器来实现

var xxx=setInterval(function(){

var stop = document.documentElement.scrollTop 这是重新获得值

stop -= 200; //自身高度-200，变量新的高度赋给原来的变量（最后）

if (stop <= 0) { //判断是否到顶，以确认是否应该确认清除定时器

clearInterval(xxx); //到顶，清除定时器

} else{

document.documentElement.scrollTop = stop;

//变量新的高度赋给原来的变量 ，继续向上滚动

}

},50);

}

## 动画原理

什么是动画：一系列的图像通过周期性的展示就可以形成动画。（时间轴）

JavaScript中的动画原理：

对象的属性（大小，颜色，旋转的角度，位置，透明度，渐变...）通过周期性的改变就可以实现动画效果。

动画的三要素：

距离distance：初始位到结束位，单位为px

时间time：动画完成的总时间，单位为mm

速度speed：动画属性改变的快慢，单位为px/mm

var circle = document.getElementById(“circle”);

console.log(circle.style.left);

取值也会涉及到兼容性问题：

IE中获得位置的值的方式：

circle.currentStyle

非IE中获得lef位置的值的方式：

var styles = document.defaultView.getComputuredStyle(circle);

解决兼容性的问题：

var eStyle = circle.currentStyle || document.defaultView.getComputedStyle(circle);

取得left的位置值：

eStyle.left 就是能在兼容性下能获得的值。

在timeInterval(function(){

像素取整：var move = parseInt(eStyle.left);

移动：move += 100

满足条件停下来clearInterval()

不满足条件，重新赋值，并拼接单位：circle.style.left+ “px”

})

## 动画的函数的封装

# 2018.02.06

## Z-index

Z-index 仅能在定位元素上奏效（例如 position:absolute;）

该属性设置一个定位元素沿 z 轴的位置，z 轴定义为垂直延伸到显示区的轴。如果为正数，则离用户更近，为负数则表示离用户更远

# 2018.03.06

## 关于html,css,js页面完成后复习

居中问题

块级元素水平居中：

margin: 0 auto; width: npx;

行内元素水平居中：

text-align: center;

多行文本居中：

display:table-cell;

vertical-align: middle;

垂直居中都用：height 和line-height

input的type属性：

text

password

submit

button

reset

radio

checkbox

file

hidden

**input的name属性：**

注：表示控件的名称，有名称的标签数据才会提交到后台

判断是否清除浮动：

控制台查看：父元素的高度是否为0，内容是否上串

定位：

static:默认，没有定位

absolute：绝对定位，相对于最近父元素。没有就以body。

fixed: 固定，相对于浏览器

relative：相对，相对于自己，

标准模式，怪异模式：

以前：使用减法，用定义的宽高减去padding。

现在使用box-sizing:

border-box: 怪异

content-box: 标准

parseInt(): 其他类型转换为number中的整数...

1. 开头是数字，才能得到具体的数字，其他都只能得到NaN
2. NaN：的typeOf还是number类型

强制数据转化。

单行语句。if

多行语句。

函数和变量优先级

函数声明和变量（未赋值）同名的情况下，函数的优先级更高。

优雅降级&渐进增强

优雅降级：对无法支持新功能的老式浏览器，增加候选方案，使之降级体验但又不会完全失效。

渐进增强：从所有浏览器支持的基本功能开始，逐步添加新式浏览器支持的功能。

浏览器的兼容性问题

不同的浏览器对同一段代码有不同的解析。而需求是，无论什么用户用什么浏览器