# 第一天 HTML第一天

## 1-1 前端开发

1.1 浏览器内核：

IE内核——trident（兼容模式）；

Firefox——Gecko

Safari——webkit

Chrome——bink

1.2浏览器内核有：渲染引擎和JS引擎

1.3 web标准（重点）：展示统一的内容，w3c（万维网联盟），包括**三层分离原则**：结构，表现，行为；

**1.4 HTML**（Hyper Text Markup Language）超文本标记语言

**1.5 固定代码骨架**

<HTML>

<head>

<title></title>

</head>

<body>

</body>

</HTML>

**1.6 UTF-8 最常用的字符集**

**1.7 标签**

1. 标题标签：h1，h2，h3，h4，h5，h6 6个等级，logo一般用h1标签；
2. 段落标签：<p></p>
3. 文本格式化标签

<b></b> <strong></strong> 后面应用较多

<i></i> <em></em> 斜体

<s></s> <del></del> 删除线

<u></u> <ins></ins> 下划线

1. 布局标签：div 和span标签
2. 图像标签：<img src=”图片的路径” alt=”图片不能显示时代替显示的文本”> title 悬浮标题，width height 设置宽高， border 设置图像边框宽高
3. 链接标签：<a href=”跳转目标”>必须有内容，否则不会显示</a> target属性控制窗口的弹出方式：\_self默认值，不保留当前的窗口，\_blank 保留当前窗口，打开新的窗口；

锚点定位：创建锚点

a <a href=”#id名>”链接文本”</a> 创建链接文本；

b 使用相应的id名标注跳转目标位置；

1. base标签 可以统一控制所有a标签的打开方式

**1.8 路径**

1.8.1 绝对路径：带有盘符的路径/完整的网络地址

1.8.2 相对路径：图片在文件的下一级目录：先进入文件夹，再找到图片；

图片在上一级目录：先返回上一级目录../；

**1.9：列表：无序列表，有序列表，自定义列表**

# css第一天

**2.1 表格（table）**：tr代表行，td代表单元格 ，标题Caption 表示，必须在table里面，表头由th代替tr;

**2.2 table 属性**（行内式）

1. border 边框
2. cellspacing 单元格间的空隙
3. Width height
4. align <水平对齐方式：left center right>
5. cellpadding<内容与单元格的空隙>

**2.3 合并单元格** ：跨行合并rowspan=”n”几个单元格，后面要删掉；

跨列合并 colspan

**2.4 表单标签-input 单标签 type属性**

1. **text 文本框**
2. **Password 密码框**
3. **Radio 单选按钮 -使用时必须有个name分组，形成组内互斥效果**
4. **Checkbox 多选框**
5. **Button 普通按钮**
6. **Submit 提交按钮**
7. **Reset 重置 这个和submit必须告诉它重置那张表单**
8. **Image 图片按钮**
9. **File 文件域（上传文件）**

**name 控件名称 ； value input控件默认文本值 ； size ...在页面显示的宽度 ；checked 定义默认被选中的项；maxlength 允许输入最多的字符数**

2.5、绑定标签 label **<label for=”male”>male</label>**

2.6 、textarea 文本域

<textarea cols=”可输入字数” rows=“行数”></textarea>

2.7、下拉菜单

<select>

<option>选项1</option>

</select>

2.8、datalist 选项列表 与input元素配合使用

<input list=””>

<datalist id=”和list名称一样”>

<option>选项</option>

</datalist>

**2.9、fieldset 与legend 搭配使用**

<filedset>

<legend>组别名称</legend>

内容

</filedset>

**3.0、表单控件新属性**

占位符 placeholder

自动获得焦点 autofocus

上传多文件 multiple

自动完成 autocomplete

必填项 requred

音量大小 range

月份 month

周 week

时间 date/time

**3.1、多媒体标签**

<embed src=”复制网址就可”></embed> 看视频

<auto src=”目标文件”></auto> 音频 autoplay 自动播放，controls 控制，loop 循环播放， source 调格式

<video src=”目标文件”></video> 离线视频

### 1-1-1 CSS显示模块及初始化

1. css的三大显示模式：块（block）：默认独占一行，可设置行高，，若没有设置宽度， 默认宽度为父盒子的宽度；

行内元素（inline）：一行显示多个，不能直接设置宽高，内容撑 开盒子。

行内块（inline-block）：可一行显示多个，也可设置宽高。可通过 display进行相互转换。

1. 层叠性；两个一样的选择器（权重一致）作用在同一个标签上，有发生属性冲突，后者覆盖前者。
2. Css初始化：body,p,div,h1,h2,h3,h4,h5,h6,input,a,ul,li,di,dt,dd {

Margin:0;

Padding:0;

}

body{

Font:12px/100% “微软雅黑”；

Margin:0 auto;

}

.w {

width:1200px;

Margin:0 auto

}

Input,img {

border:0 none;

outline:none;

Vertical-align:middle;

}

a {

text-decoration:none;

}

Li{

List-style:none;}

.Clearfix:after {

Content:”.”;

Display:block;触动bfc

Height;0;

Visibility:hidden;

Clear:both;

}

.Clearfix {

Zoom:1;

}

## 1-1-2 CSS浮动布局

### 浮动布局（float）：

1. 特点：a元素浮动后不占据标准流（原来）的位置（脱标），会把后面的标准流元素（顶上来占据脱标盒子的位置）覆盖，子盒子在父盒子区域中浮动。

b浮动的盒子在一行显示，浮动的元素只能在标准流后面浮动，不能占据标准流的位置，且浮动元素默认顶端对齐；

C 行内元素在浮动后可设置宽高，即转为行内块。但不推荐使用。

1. 作用：a文字环绕（最开始的作用） b制作导航 c网页布局
2. 清除浮动：清除浮动产生的影响，方法如下：

a 额外标签法：在最后一个浮动元素后面加一个额外的标签，并且给予它属性<.clear{clear: both}>

b 给父盒子设置高度；

c 给父盒子设置overflow：hidden，为触发块级格式上下文，盒子不影响外部；

d 伪元素清除法，（利用伪元素的特性，可以实现添加额外元素但不影响结构代码）方式见上页3最后；

e 双伪元素清除法: .clear::before，.clearfix::after {

Content:”.”;

Display:table;}

.clearfix:after{

Clear:both;

}

.clearfix {

Zoom:1;

}（可省略）

1. 浮动的影响：当一个标准流的父盒子中的子盒子全部浮动时，会导致父盒子失去高度（因为父盒子是由子盒子撑开，子盒子浮动时会脱标。）

## 1-1-3 CSS定位

1. 盒子定位（position）
2. 静态定位（static）：默认值就是标准流；
3. 绝对定位（absolute）：元素使用后不占据原来的位置（脱标）；

位置从哪里出发视父盒子的情况而定；当与父盒子都使用绝对定位/相对定位/固定定位时，外置参照父盒子进行位移；

可将行内元素转变为行内块元素，但不推荐使用；

3、相对定位（relative）：元素从原本在标准流中位置出发，还占据之前的位置，不会转换显示模式。父盒子使用绝对定位，子盒子使用相对定位（父绝子相）用的最多，位置从父盒子出发。

4、固定定位（fixed）：位置从浏览器可视区域出发 ，会转换显示模式为行内块，可以脱离标准流，不占据位置，可用方位名词来控制位移。

改变定位的盒子层级显示：z-index：数值； 值越大，层级越高。子元素的层级永远高于父盒子。

层及顺序：标准流<浮动流<定位流！

1. 利用定位画简单的图形
   1. 三角形：w:0;h:0;border-style:solid; border-width:50px ; border:red transparent（透明）四条边都要写；

### 三、css的高级样式

1. 鼠标样式（cursor）：在标签/style中添加，样式有：pointer 小手；default 箭头；

Move 十字移动，text 默认文本 ；wait 等待。

1. 防止文本拖动 Textaera{ resize:none;}
2. 文字溢出 ，多出的字由...代替：text-overflow ：ellipsis（文本溢出时省略号代替）必须跟overflow：hidden；文字强制在一行显示white-space: nowrop; 一起使用；
3. 元素显示和隐藏的三种方法：overflow 溢出隐藏 ；display 消失；visibility 隐藏;
4. Transition 过渡：有四个属性，如果都写的话，写一个all就可以了，

/\* transition: 要过渡的属性 花费时间 运动曲线 何时开始; \*/

transition: width 0.6s ease 0s;速度分为：linear匀速，ease逐渐慢下来，ease-in 加速，ease-out减速，ease-in-out先加速后减速；

6、移动translate（x,y）

### 7、缩放 scale(x, y)

8、旋转rotate（deg） transform:rotate(45deg);

9、调整元素转换变形的原点，div{transform-origin: 10px 10px;transform: rotate(45deg); } /\* 改变元素原点到x 为10 y 为10，然后进行顺时旋转45度 \*/

四、京东项目：

**项目流程注意事项**：logo一定要用h1标签包着，代表是整个页面最重要的元素。

# JS就业班笔记

## 1 JS就业班第一天

### JS-行为标准

1.1 组成部分：ecmascript-js标准

Dom 通过js操作网页元素

Bom 通过api接口操作浏览器

1.2 作用：1.页面特效；

2.移动端；

3.异步与服务器交互；

4.服务端开发；

1.3 特点：简单易用、解释执行、基于对象。

1.4 js书写位置:

1.4.1 内嵌式：可写在页面的任何位置上。推荐写在html结束标签后边。<script type=”text/javascript”> </script>

1.4.2 外链式：<script src=”js路径”> <script>链接进文件。中间再写内容是无效的，当浏览器解析时会把外联文件中的内容替换到此处。

1.4.3 行内式：onclick=”alert(“”)” 在dom中使用。

1.5 快捷键常用的使用：Ctrl+D 复制； Ctrl+Y 删除行； Ctrl+alt+L 自动格式化代码文件；

1.6 JS中的注释（内容被浏览器跳过）：// 单行注释； Ctrl+shift+/（Ctrl+/） 多行注释

1.7 转义符：\”输出双引号 换行 \n 多打几个空格 \t

### 页面输出消息的方式；

1. alert（“”）在页面中弹出对话框，里面只能写直接量和变量。

1.1 直接量（数据类型）: 字符串，数字，bool;

1. Confirm（“”）在页面中弹出对话框,比1多了确定与取消按钮。
2. Console.log （“”）在控制台中输出信息。
3. Prompt（“”）弹出对话框，用于接收用户信息。
4. Document.write（“”）在页面输出信息，且能输出标签，慎用，可能会覆盖原来HTML的元素。

### 3、变量

3.1 定义变量：在内存中开辟一个空间，用来储存数据；

3.2 变量定义：用var定义，可以同时声明多个变量，中间用逗号隔开。

3.3 列子：var wukong =”123”

alert(wukong)

3.4 变量的命名：1.不能以纯数字、数字开头定义变量名，

1. 不允许出现空格和其他符号，中文命名可以，但不推荐，保留字和关键字也不推荐使用；
2. 只能用$ 和 \_和字母开头定义变量名，后面接任意的数字、下划线、字母和$符号。

3.5 变量名的约定：1、起名要有意义，语义化，最好与保存的数据的作用有对应关系。

2、使用驼峰命名法；第一个单词的首字母小写，后面任何单词首字母大写；

### 数据类型

4.1 number-数字类型（正、负、小数）

表达方式有：十进制 ，十六进制（已0X开头，数字用0-9，A-F、a-f表示），八进制（以0开头。用0-7表示），二进制（0b表示）。

**4.2 判断数据类型 用Type of（变量）判断变量属于什么类型。**

4.3 字符串类型-String：必须用单引号/双引号括起来 var=” ”,主要用来保存和显示文字;

4.4 布尔类型：只有两个值 true 和 false 。其中true=1，false=0；

4.5 undefined: 变量没有赋值，即变量未初始化；

4.6 null: 值为空，变量为引用，是一种对象类型，变量只能被赋值为null，不会自己变成null。

**4.7** 算术运算符：a：+ 两个数字类型的数相加，得到数字类型；1个数字和1个字符串相加，得到字符串类型2个也一样（后半部分作为连接符），布尔类型在数据上来讲还是数字，true=1，false=0；

b: - 1个数字 一个字符串/两个数字相减，会尝试把字符串变成数字，均得到数字类型；非数字类型的字符串相减，得到NAN（not a number）也是一个数字类型。\* /号一样。

C: 除号，1个数字 一个字符串/两个数字相除，均得到数字类型；档0作为除数时，结果为infinity 无限大，是一个数字类型

d : % 取余，与0取余，得到NaN。

**备注：== 判断两个变量是否相同，不判断数据类型；=== 两者均判断；**

**！= 不等于**

4.7.1 待操作的数据运算式：

a += a+=b a=a + b

b += a-=b a=a - b

c \*= a\*=b a=a \* b

a /= a/=b a=a/b

4.7.2 逗号表达式：表达式用逗号隔开，前面的表达式都被执行，只是结果为最后的表达式的值var res=(n1++，n2++，n1+n2）；。

**4.8 优先级**

**1，括号（）； 2，！ - ++ -- +； 3，\* / %**

**4, + - 5, < <= === !==**

**6, & 7,|| 8，三元运算 9， = += -= \*=;**

**作业：**1） 【难度★★】用JS计算下列算式，并将结果在控制台输出：



2） 【难度★★】世界上表示气温有两种方法：摄氏度和华氏度。两者的关系是：



编写程序，让用户输入摄氏度，向用户提示出对应的华氏温度。

不做要求

3） 【难度★★★】用JS计算下列算式，并将结果在控制台输出：



Math.pow(5,7)

下面的所有不做要求

4） 【难度★★★】编写程序，提示用户输入六边形的边长，然后显示它的面积，计算六边形面积的公式是：



Math.pow(3,1/2)

5） 【难度★★★★】如果今天是星期六，那么1000天后是星期几？用一条语句计算出来。

6） 【难度★★★★★】用户输入一个三位数，用程序计算三位数字的和。

比如：

用户输入155，就弹出11

用户输入316，就弹出10

用户输入989，就弹出26

用户输入678，就弹出21

注意：用户输入数字的有效性、合法性不用考虑。比如，如果用户输入34343、“我爱你”，这种情况我们不要去考虑，你就想像，用户会非常乖，一定会输入3位数的。这是因为我们没有学习if语句。

abc

得到第一个数：Math.floor(abc / 100);

得到第二个数：Math.floor(abc/10)%10;

得到第三个数：abc%10

## 就业班第二天-date对象和math方法

### Date对象方法

1.1 getDate（）从Date对象返回一个月中的某一天（1-31）。

1.2 getDay（）从Date对象返回周中的某一天（0-6）0为星期天。

1.3 getMonth（）从Date对象返回月份（0-11） 0为1月，11为12月。

1.4 getFullYear（）从Date对象以四位数字返回年份。

1.5 Date 返回一个完整的时间。

1.6 getHours 小时

案例：

Var mydate=new Date(); 取得当前时间

alert（Date（））；

alert （mydate.getDay（））；

Alert （mydate.getMonth（）+1）

### Math对象-执行数学对象

2.1 ceil（X） 对数进行上舍入（天花板函数）>=X;

2.2 floor（x）对数进行下舍入（地板函数）；

2.3 max（x ,y) 返回x ,y 中的最大值；

2.4 min（x ,y) 返回x ,y 中的最小值；

2.5 pow（x,y）返回x的y次方；

2.6 random（）返回0-1之间的随机数；

2.7 round（）四舍五入，只看小数点第一位数。；

案例：

Var n1=12；

Var n2=1.23；

Var n3=3.98；

Var n4=Math.ceil（）；

Alert（n4）；

### 进制转换

值=位值\*进制的（位置-1）次方+.....

### 数据类型转换

4.1 数字转字符串：a **变量**.toString（），null和undefined不能转，NaN可以转； b String（）

4.2字符串转数字类型：

**a Number**（）非数字型或数字中间有空格的结果是NaN(not a number） 是一个数字类型。Null转换为0；

**b parseInt**（）非数字型的字符串（只保留前面的数字），只识别前面的数字，后面的无法识别，没有数字或第一个为非数字时的为NaN；

小数时，相当于取整（工作原理：从左至右依次来转，知道遇到第一个非数字）。

**c parseFloat**（）数字开头的只保留开头的数字；小数保留原数据；非数字型的结果是NaN(not a number） 是一个数字类型。

4.3 转布尔类型：Boolean （）

A 数字、字符串、Infinity，除以下转变为false的几种立项，其他的转完为true；

B 0、null、undefined，NaN ，空字符串 转为false。

4.4 隐式转换：不需要调用任何方法就能转换，只要参与运算就可触发。1、在字符串前面写+、- 会尝试转换为数字，非纯数字只能转为NaN。2、直接参与+外的数学运算。

4.5 isNaN 判断一个内容不是数字，不是数字结果为true，反之为false，如果字符串里面只有数字，返回false，作用是判断数字是否转换成功

；

### 逻辑运算：true 、false（两个值），强制转布尔类型。

1、逻辑运算符：

5.1 &&且 ：条件均为真才为true，有一个错就是false。找到false就返回，结果就是false的值，如果都是true取后面的值。

5.2 || 或：条件有一个为真，就为true，找到正确的就返回，都是true取前面得值。

5.3 ！非—取反 ：真变假。假变真。不是0的数都为true,取反为false。

1. 等号运算符：

2.1 ==判断内容是否相同，不判断数据类型。

2.2 = 赋值等。

2.3 === 判断数据类型，且判断内容是否相同。

2.4 ！= 判断内容是否不同，不判断数据类型（不等于）。

2.5 !== （不全等于）不仅判断内容是否不同，且判断数据类型。

1. 逗号运算符： 用“，”进行变量的集体声明。

## 就业班第三天-选择结构，语法

结构：a 顺序结构 b 分支结构（登录） c 循环结构

### 1、if...else...语句

属于分支结构-if语句。

If （条件表达式）{

如果表达式为true，执行该处代码。

}else {

如果表达式为flase，执行该处代码

}

### if...else嵌套

判断多个条件

If(条件表达式){

如果条件表达式结果为true,执行该处代码。下边代码不执行。如果为false,执行下边代码。

}else If(条件表达式){

如果条件表达式结果为true,执行该处代码。下边代码不执行。如果为false,执行下边代码。

}else If(条件表达式){如果条件表达式结果为true,执行该处代码。下边代码不执行。如果为false,执行下边代码。

}else{

如果上边条件表达式结果都为false,执行该处代码。

}

### 3、三元表达式：表达式？结果1：结果2；

案例：var n1=prompt（请输入数字）；

N1%2==0；

alert （“奇数/偶数”）

N1%2==0？alert（偶数）：alert（奇数）；

若表达式正确，则执行结果1，反之为结果2。

### F12代码调试工具；

代码调试：代码（刷新）查看—设置找到Source—找到调试的文件—设置断点—刷新——选中变量名，不能添加固定的变量，右键Add to Watch,点击眼睛图标进行执行，每点一次执行一步。

### Switch

1. （条件过多，可优先使用；其比较用的是===，不仅要值相同，且数据类型要一样。也可以做集体声明）

也是分支结构

Switch(变量){

case 值1:

如果变量和1的值相同（全等），执行该处代码

break;

case 值2:

如果变量和2的值相同（全等），执行该处代码

break;

case 值3:

如果变量和3的值相同（全等），执行该处代码

break;

Default（位置没有要求，不影响功能）:

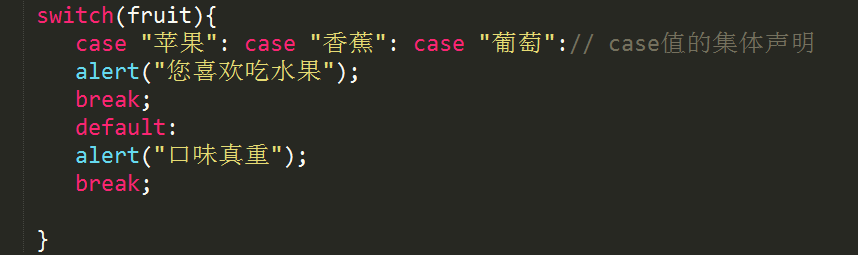
如果变量和以上的值都不相同（全等），执行该处代码

break;

}

**★switch后边的变量和case后边值的数据类型必须保持一致。**

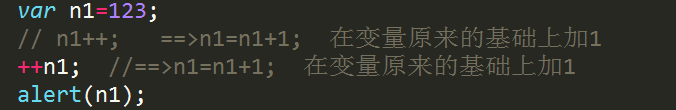
◆Switch语句可以对变量进行集体判断：



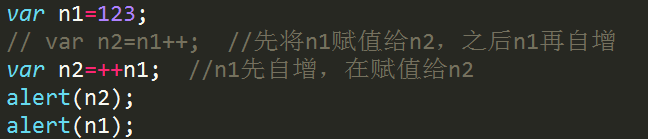
### 自增自减

i++ ++i

◆在不参与运算的情况下，i++和++i都是在变量的基础加1



◆在参与运算的情况下



### while循环（重复循环一段代码，直到某个条件不在满足）

While(条件表达式){

只要条件表达式结果为true,循环一直执行，当条件表达式结果为false的时候，循环终止

}

While循环语句需先在循环体外定义变量。先判断后循环。

7.1 使用时间：如果不确定循环次数/本身是一个死循环，用while；

### DoWhile

Do{

循环体代码；首先执行该循环体代码一次。如果while后边的表达式结果为true,该循环体会一直循环。如果结果false,该循环终止。

}while(条件表达式)

先循环后判断，比while多循环一次

### For循环

for(定义变量;条件表达式;自增自减){

循环体代码

}

执行顺序:

◆首先定义变量

◆如果条件表达式结果为true的时候，执行for循环里的代码，如果为false，循环体代码终止执行。

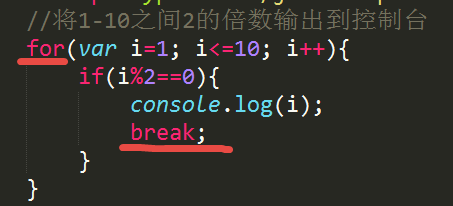
◆先执行变量和条件表达式循环一次，再执行自增自减。

◆ 哪怕里面的内容都不写，分号也不能省略；

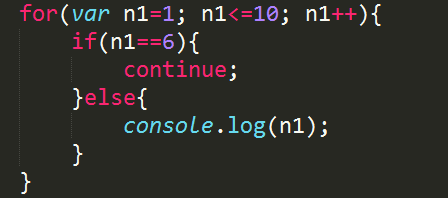
取到最小数为：-Number.Max\_value

### Break

### 在循环体内，只要代码遇到break,程序**立马结束当前循环**。当前循环指的是break语句所在的循环体。



### continue语句，跳出本次循环该语句后面的代码不再执行,整个循环体继续循环。



1. 循环注意点：

1、避免死循环，记得写i++；

2、避免永真永假；

## 就业班第四天-数组

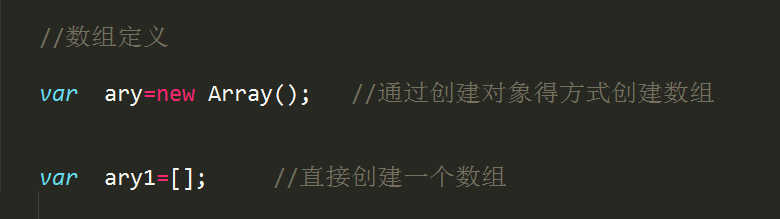
### 定义

数组名称和变量命名一致

1.1 元素：数组中保存的每个数据；

1.2 长度（arr.length）：数组中保存元素的个数；下标=长度-1

☞如果不按顺序添加元素，中间会自动补元素，元素的值为undefined；



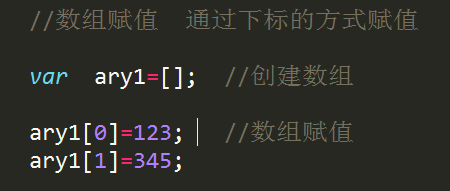
### 如果在new Array（）括号中只写一个数字如10，代表数组中有10个元素。元素值只有undefined

1.3 验证是否是数组：1；xxx instanceof() 还可判断其他类型；

2；Array.isString（arr）

### 2、赋值

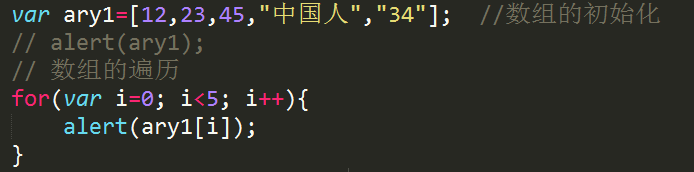
☞数组中通过下标的方式进行赋值。下标从0开始。



**☞此案例中的数组是通过字面量的方法创建的数组。**

☞如果在arr1（）括号中只写一个数字如10，代表数组中第0个元素。元素值为10；

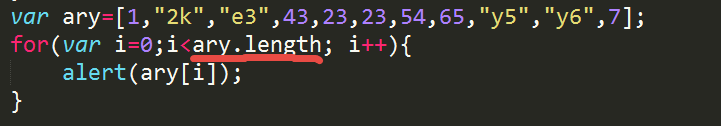
### 3、数组的初始化和遍历

4、数组数据的个数 length属性（可变化，变小后面多余的元素删除；变大后添加undefined补齐）

4.1 给数组添加元素的方法：

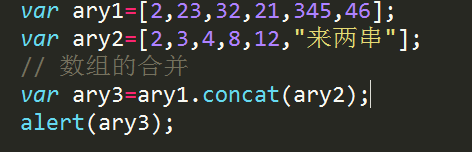
1、通过索引进行添加

2、通过数组名.length获取数组长度（元素个数），可以按顺序给数组加长度。



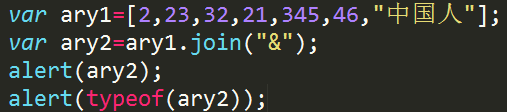
### 5、数组合并

使用concat方法合并数组。



### 6、Join 方法 返回一个字符串

Join方法返回一个字符串数组。



### 清空数组：

A arr.length=0;

B splice(0);

C arr=[]；

1. **闭合原则**：用于去掉数组中重复的元素，和找到素数；作用就是将比较全部完成后，再进行进一步操作；
2. 二维数组：在数组中的元素也是数组；

### 数据类型

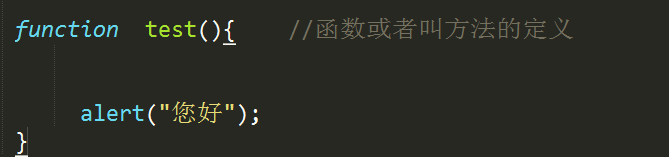
10.1基本数据类型：string，number，boolean,undefined,null；值在栈区（容量小），先进后出，只传值。

10.2 复杂数据类型：数组，function，Object，值在堆区（容量大），先进先出，会在栈区开辟保存一个地址，传递的是地址而不是真正的数据；（面试题）

## 5、就业班第五天-函数

### 1、函数（方法）定义/声明：

通过 **function**  关键字，和自定义方法名（**函数名**） 既可定义一个函数。 如下：



方法的创建，函数声明不能执行，只有通过调用方法（函数名（））调用后才可以执行。

### 函数的组成：

Function fn(a,b){

函数体（要打包的代码）

}

**2.1 参数**-参加计算的数值（就是调用者给函数传递数据用的）；

2.1.1 分为形参（a,b不参加计算，给实参占位置）和实参（调用时，实际参与计算的数值）；

2.1.1.1 形参个数=实参，正常计算；

2.1.1.2 形参个数<实参,正常执行计算，多余的实参不使用。

2.1.1.3 形参个数>实参 看程序是否报错（报错、NaN、undefined）

A 任何值与undefined运算，结果都是NaN；

B 未给定参数的形参进行强制执行，会报错。

2.1.2 作用：增强函数的功能性；与程序员的交互性；以及和函数的可拓展性；

**2.2 返回值**（必须要执行函数才能得到）；

2.2.1 a执行完毕函数以后，能给其他函数赋值；

b 有return就有值，没有return就没值（undefined）;

c想把函数内部的值赋值给外面，必须使用return；后面也可以不加任何值=undefined

Return 切断函数 break 跳出循环 continue跳出本次循环，进入下一次循环。

2.2.2 作用：函数出来的数为半成品，需要进行外部进行二次加工。

**2.3 功能**；

**2.4 匿名函数**：没有名称的函数

（Function（）{函数体}），外加小括号，不报错。

作用：调用和自调用；

事件绑定；

定时器：a 循环进行：set Interval（Function（）{函数体}, 1000）。

B 定时进行：setTimeout(Function（）{函数体}, 延迟时间）。

**2.5作用域：变量或函数作用的范围区域；**

**2.5.1全局作用域：script开始到结束的函数外的所有部分——0级作用域链；**

**2.5.2局部作用域：每声明一个函数，函数大括号里就是；**

**全局变量：**在函数外部定义或者进入script后立刻定义的变量，以及没有var的变量；

**局部变量**：在局部作用域定义的变量，只有在局部被访问到。

2.5.3不规则声明变量：不加Var声明变量；相当于全局变量，写在局部区域时，函数不调用，就没有；

**2.6 访问变量的规则--就近原则：**先在同级作用域进行访问，优先使用同级中的要找的变量，没有再逐个找上一级作用域，没有就报错；

**2.7 预解析:把所有作用域里面的变量声明和函数的声明，提前到他所在的作用域链的最顶端，同意作用域里面，函数的声明可以在任何位置，都可以被调用；只会提升真正声明的变量，不规则的不会提升，还是按正常顺序解析；**

**2.8 函数说明：/\*\*+回车，**可以进行函数功能说明；

## 6、就业班第六天-对象（是具体的）

### 1、对象类型：就是程序中用代码描述生活中一个具体事例的数据类型；

1.1 面向对象：可以创建自定义对象，很好的支持继承、多态；

1.2 基于对象：和面向对象相反，只能new出object对象（var hero=new Object（））；

1.3 对象的属性（对象的变量/任务特征）和方法（对象中的函数/技能）/状态和行为；

案例：var hero=new Object（）创建一个空的对象类型

hero.money=10000; 属性=特征

hero.level=6;

行为=方法 hero.attack=function（）{

Console.log（“攻击了水晶！”）

}

Console.log(hero);

缺点：创建单个对象还好，创建多个函数时就会很繁琐，且for循环不能修改名字的值。

1.3.1 调用属性：对象名.属性名 console.log (hero.money)

1.3.2 调用方法：hero.attack(); 调用自己的属性：this.money

1.4 多个对象的创建：创建一个函数.精简版

function createStudent{

Var student=new Object（）；

Student.name=aaa;

Student.sayHi=function(){

Console.log(this.name+“说：大家好”)

}

return student;

}

Var student1=new student（“李四”）；

student1.sayHi()； 调用方法

1.5 创建空白对象：obj=new Object();

1.6 创建自定义对象：

1.6.1 **this**：只出现在函数中，谁调用就代表谁；

1.6.2**new**：开辟内存空间，存储新**创建的对象**，把this设置为当前对象，执行内部代码，设置对象属性和方法，**返回新创建的对象**。

**终极版：利用构造函数来自定义对象 \*\*\*名字首字母要求大写**

function Student（name）{

this.name=name; 属性

方法 this.sayHi=function(){

Console.log(this.name+”说：大家好”）；

}

} 调用与上面一致;

1.7 属性绑定：对象名.属性；对象名[对象]；对象名[值]；

1.8 进制转换：3.8.1 parseInt（数值,进制）一个参数就取整，两个参数转为**十进制，吧数字转换为10进制**；

3.8.2 toString() 无参转换为字符串，带参进制转换为：十进制转其他

Var num=7;

Var renyi=num.toString(2) 转2进制

Console.log（renyi）;

## 2、递归：就是函数自己调用自己，类似于我们的计数器，开闭原则；必须有跳出条件，否则为死循环，计数器：在函数之前保存变量之前的状态。

## 回调函数

一般用于定义一个规则来使用，规则的传递通过变量无法实现；函数作为参数进行传递和使用；

Function fn(demo){

Demo()=执行函数=函数名（）；

}

定义规则案例: 加法

Function test1(a,b){

return a+b;

}

fn(a,b,test1);

1. **对象字面量**：就是一个**大括号{}** 里面的属性和方法为键值对，之间用“，”隔开；可通过Var s1={}创建一个空对象，中间也可以有值。
2. **Json（轻量级数据交换格式，不包含方法）** 与上面的区别在于键值对中的键必须要有引号，对象本身没有length，所以不能用for循环进行遍历，只能用for...in...

Var json={“name”:拴住，“age”:0,”arr”:[1,2,3]}

Console.log(json);

For (var k in json){

Console.log(k);k表示属性键

Console.log（json[K]）;代表k对应的值

}

Json.k 代表k属性，不能代表对应的属性值，取值一般用json[K]来去，json.k也可以单不推荐；

参数赋值的时候，简单数据类型传值，复杂数据类型传地址值；

## 4、内置对象方法-API

主要学习API，H5的方法。

API： 参数，返回值，功能，调用者；

4.1 数组的创建方式：a；var arr1=[1,2,3];

B; var arr2=new Array();

4.2 instance of 判断是不是数组/函数类型；

Console.log（arr instance of Array）;

4.3 转换数组: 数组.toString（）数组转字符串；

Value Of（）返回数组本身；

4.4 join：把数组元素用特殊方式连接变成字符串；

Var arr=[1,2];

Var str1=arr.join（）无参数用，连接；

Var str2=arr.join（“ “） 元素间有一个空格；

Var str3=arr.join（“”）空字符串，无缝连接；

4.5 push：在数组的最后增加元素并返回新的数组：

Var arr=[];

Var arr.push(要增加的内容);

Console.log(arr);

4.6 pop:取出数组的最后一个元素并返回新的数组；

4.7 arguments--参数（是个伪数组，保存的是调用函数时传递过来的所有实际参数）:只在函数中使用，在函数在外使用会报错，主要是用来间接实现函数的重载（函数名相同，但函数的参数个数）；JS没有重载，可以利用arguments进行重载；

Fn(1,2) [1,2]

Function fn(a,b){

Console.log(arguments);

}

4.7.1 伪数组：无法直接调用数组方法或期望length属性有什么特殊的行为，不具有push，pop等方法。

4.7.2 伪数组转换为数组的方法：伪数组.from({});

4.7.3 **argument.callee**,代表整个函数

4.8 shift（）：取出数组中的第一个元素，返回新数组；

4.9 unshift():在数组前面提前加一个或多个元素，并返回新数组；

4.10 排序方法：6.10.1：翻转arr.reverse（）；

6.10.2：排序 arr.sort（）根据字符从小到大排列，可根据回调函数一起使用。

4.11 concat 连接，拼接；将两个数组凭借成新数组；arr3=arr1.concat(arr2);

4.12slice 数组的截取：arr.slice(开始位置，结束位置），开始位置=0.代表从数组第一个数字进行截取；开始位置为负数，代表从数组尾部开始截取；

4.13 splice删除、替换元素（原数组），arr.splice(开始位置，截取个数）；

4.14 split 将字符串截取成数组，可自己定义截取的位置；

4.15 indexOf 从前到后给元素查索引；lastindexOf则相反；

# WebAPI第一天

1、API（接口）-系统提供的封装好的函数；

2、JS组成;

1. Dom-文档对象模型；
2. Bom-浏览器对象模型；
3. 核心-ECMAScript；

3、获取元素的方法：

1. 通过id获取：要获取的元素=docunment.getElementById(“元素名”)；只能用document来调用
2. 通过类名获取：要获取的元素=docunment.getElementsByClassName(“类名）；
3. 通过标签名获取：要获取的元素=docunment.getElementsByTagName(“标签名’）；
4. 后面两种方法的得到的是**伪数组**，可通过索引值得到其相关想要的元素；

e. 拿到元素的**行内的**相关属性：点语法：元素名.属性名；中括号也可以获取，也可修改和赋值；

g. 拿到标签里的文字：对象名.innerText，也可赋值；

1. script写在body上面需要加载函数：

window.onload=function(){事件}；代表图片或页面加载完毕后调用；

5、querySelector（selecteror）查询传入Css的选择器来找到元素，需要传入字符串，此方法只会返回一个元素，如果有多个，返回第一个；要找全部queryselectorAll（selecteror）；

6、事件三要素：

a. 事件源-执行者，谁引发/谁交互谁就是事件（获取事件源和相关事件）；

b. 事件-执行的事情（绑定事件）；

c. 事件驱动程序（匿名函数，对样式和HTML的操作）；

7、Dom（由节点组成）概述：

解析过程：HTML加载完毕，渲染引擎会在内存中把HTML文档生成DOM树，getElementById获取树的元素，操作的时候修改该元素的属性。

8、作业：

6.1点亮盒子（使所有的盒子都没有此属性，只有自己赋值属性排他思想，与for循环连用）

6.2 禁用和解禁：给disable赋值true/false；

6.3 一个按钮实现禁用和解禁；

6.4 开关灯案例；背景色变黑变白；同理隐藏和显示盒子；

6.5 模拟京东搜索框，获得焦点：onfocus，input value=“”；失去焦点：onblur; placeholder属性可以得到同样的效果；

6.6 显示和隐藏二维码：onmouseover 和onmouseout,鼠标移入和移出事件；

6.7 隔行变色并且鼠标移入后变色，移出后变回原来的颜色；

# 8、WebAPI第二天

1、节点

1. 节点获取

a 直接获取

b 访问关系获取：父节点进行操作；

1. 节点类型
2. 父节点parentNode
3. b.兄弟节点sibling 上个兄弟节点：节 点.previousElementSibling||节点.previousSibling，Element 可写可不写，如果不写element可能获得文本节点；下个节点将previous 更换为Next即可；
4. C. 所有子节点：childNodes：获取所有的节点，包括文本节点，如换 行；Children:只得到元素节点，用的多,IE8之前也会得到注释节点。
5. d. 第一个子节点：firstElementChild||firstChild；
6. e. 最后一个节点：lastElementChild||lastChild；
7. f. 随意得到兄弟节点：节点自己.parentNode.children[index]
8. 获取body：可以给body加id、类名、**ducument.body;**
9. 节点操作：
10. 创建节点：新标签（节点）=docunment.createElement(“标签名”)，内容另加；要配合appendChild（）使用，利用dom的API创建元素

Document.write()有缺陷，能解析标签，但在事件里会覆盖原来的全部内容，不常用，只能给body加；

innerHTML 能解析标签，也会覆盖之前的内容，如果要保存之前的内容就用innerHTML加等于要加的标签，工作中用的多；

1. 添加元素/插入节点：父节点.appendChild()在后面插入；

父节点.insertBefore(要添加的标签，插入到哪个子元素前面)在某个元素的前面插入；

1. 删除节点：父节点.removeChild（子节点）；
2. 复制节点：新节点=被复制的节点.cloneNode(参数），参数为true,为深复制，无参数为浅复制，只能复制一个节点，包括属性，但不包括子元素；
3. 替换元素：replaceChild（替换元素（newchild），被替换的元素（oldchild））,通过父元素替换；
4. 节点属性的设置、获得和移除
5. getAttribute（“名称”）得到相对应的值，获取所有的属性，包括自定义属性；
6. setAttribute（“属性名”，属性值）设置属性；
7. removeAttribute（属性名） 删除属性；
8. 节点关系

nodeType-节点类型：标签/元素节点：ele nodeType节点值）=1，nodeValue是null；

属性节点：id/class等 nodeType(节点值）=2；

文本节点：txt nodeType(节点值）=3；

注释节点：nodeType(节点值）=8；nodeName=commend;

Document: nodeType(节点值）=9;

2、获取和修改文字

1. innerHTML:获取双闭合标签里面的内容包括标签（识别标签）,还可以赋值/修改文字，覆盖之前的内容---**无兼容问题**；
2. innerText(微软创建的，IE曾经有兼容问题）：获取双闭合标签里面的内容（但不识别标签），修改文字，覆盖之前的内容；
3. Value：标签的属性；
4. textContent属性，IE8及之前的不支持；
5. 阻止a标签的跳转
6. 给事件函数里面return false；

2） href=“javascript:void(0)”用的更多；

4、作业：

1. 全选，全不选，反选；
2. 关闭广告，点击a标签，关闭盒子（盒子的display变成“none”）；
3. 美女相册（点击a链接，将a连接的地址赋值给图片的src属性）；

4、Tab栏切换（点亮盒子，添加类，当点击/移入li标签时，对应索引的span标签也跟着显示，其他的隐藏）；

# 9、WebAPI第三天

## **1、字符串的部分用法（点语法）**

1）length:获取字符串的个数，包括空格；

2）trim：可以去掉字符串两边的空格，返回新的字符串，不会影响之前的字符串；

3）indexOf(字符)

A 从前到后查找字符是否在字符串中，在的话返回在字符串中的索引（单词返回首字母的下标，后面重复的不找），没在的话返回-1；

B 严格区分大小写；

C 找空字符串，返回0；

1. charAt(“索引值”)，给索引查字符。

2、键盘弹起事件：

1）onkeyup：键盘弹起事件；

2) onkeydown:键盘按下事件

# WebAPI第四天

1、设置计时器：

1. setInterval(code,delay)：code可以是string/function，delay是设置延迟时间，按时间循环执行。注：传函数名，不能加括号，加括号得到的是函数的返回值；
2. setTimeout(code,delay)，用法与setInterval一样，但只执行一次，可理解为定时执行。
3. clearInterval(number) 清除计时器，在函数计时器开始之前清除，timer为计时器编号；
4. 多次点击计时器事件，会触发多个计时器，要防止这个bug,最好在开启计时器之前先清除计时器/在点击一次以后禁用这个点击按钮。
5. 旋转：transform;retate(deg);

Deg为正值，顺时针旋转。

1. offset家族
2. offsetWidth 可以真实检测盒子的宽；包括border+paading，可获取css中的样式，为number类型；
3. 与style.width区别在于，style.width不包括border+padding，且只能取行内的样式，不能取css中的样式；
4. style.width可读写，获取的值是string类型；offsetWidth只可读，不可设置；
5. offsetHeight 可以真实检测盒子的高；包括border+padding
6. offsetParent：其实就是获取到某个元素参照的定位元素是谁；
7. 不管是不是子元素，只要是固定定位，因为它参照窗口可视区域，得到的是null.
8. 如果子元素为非固定定位，并且父元素也有定位，那么得到的是父元素；
9. 如果子元素为非固定定位，并且父元素没有定位，那么得到的是body；
10. 如果子元素的子元素为非固定定位，并且父元素没有定位，那么就一直向上找，一直到找到离自己最近有定位的元素，如果父元素均没有定位，得到body；
11. offsetLeft/offsetTop 检测盒子与定位的父盒子之间左边/上边的距离，包括父盒子的padding，不包括border；
12. BOM操作和window对象
13. window对象-是JS的顶级对象，包括dom和bom
14. 包含这个页面的所有信息，dom，history,location
15. 声明的全局变量和函数，其实就是给window这个对象加了属性和方法；
16. 默认有一个name属性。这个属性取到的永远是字符串
17. window的方法：
18. open() 打开 （url(要打开的网页地址) 、name（和a标签中的target打开方法一致）、replace、specs）
19. Close()关闭当前页面
20. onload onunload(页面关闭之后加载）onBeforeload(页面关闭之前加载）；
21. Location：host（域名+端口），hostname（域名=IP地址，网址），port（端口）等 方法：assign 加载新的文档。reload刷新
22. History 历史 history go（数字） history back() 后退 history forWard ()前进
23. Navigator 保存的是浏览器的相关信息，包括版本号，系统版本号

# 11、webAPI 第五天

1、动画匀速移动原理

1. 先给步长，步长通过目标位置与盒子本身位置进行比较，来决定取正或取负；
2. 动画运动原理：盒子位置=盒子本身位置+step+“px”；

Box.style.left=box.offsetLeft+step+“px”；

1. 在用计时器前先清除计时器，防止每点击一次开启一个计时器；
2. 在运动之前先判断目标位置与盒子本身位置的距离，在后续中用绝对值与step的绝对值进行判断，如果具体小于/等于步长，让盒子直接停在目标位置；

# webAPI 第六天

1. 缓动动画移动原理
2. 缓动动画原理（越动越慢，步长越来越小，效果更逼真）

Step=(target-box.offsetLeft)/10;

Box.style.left=box.offsetLeft+step+”px”;

1. 最后10米要注意进行判断，如果step>0，向上取整，否则向下取整；
2. 获取元素的样式属性
3. style这个属性只能获取行内的样式；
4. Window.getComputedStyle（传入想获取到样式的元素，null）,null一般用来获取伪元素；案例Window.getComputedStyle（box，null）.width;不包括边框和padding

既可以获取css中的样式，也可以获取行内的；返回的是字符串，有单位会带单位；

B (标准语法上，不能写符合属性），

C IE 8及之前不支持，IE适用于box.currentStyle.width但其他浏览器均不支持；

# webAPI 第七天

1. scroll家族
2. scrollWidth ，获得的是自身内容的宽度和高度，不包括border和margin，**包括padding，只读**；
3. scrollHeight
4. 不包括border和margin，包括padding；
5. 有一个特点，若文字超出盒子，高度为内容的高度，IE8以下，不包含IE8为盒子本身内容.
6. 获得数据**只读**，不可更改；
7. scrollLeft ,水平方向被卷去/隐藏的左边的内容像素数，可读写，存在兼容问题
8. scrolltop ,垂直方向被卷去/隐藏的内容像素数，可读写。存在兼容问题
9. **Onscroll**,正在滚动事件，页面滚动（window.onscroll）在一个页面中只能用一次，
10. 获取页面滚动的距离兼容性
11. document.documentElement.scrollTop,IE和火狐，谷歌都没有兼容性问题，**用法**和**文档说明**有关
12. Document.body.scrollTop
13. 解决方法；

Function getPageScroll(){

return{scrollTop:window.pageYOffset||document.documentElement.scrollTop || document.body.scrollTop||0;

scrollLeft:window.pageXOffset||document.documentElement.scrollLeft || document.body.scrollLeft||0

} 利用对象的方法进行传递

}

1. client家族
2. clientWidth:获取本身区域的宽和高，不包括border和margin。
3. clientHeight：获取本身区域的宽和高，不包括border和margin，只读属性;
4. **主要是为了取可视区域的宽和高；**

Function getClientSize(){

return{clientWidth:window.innerWidth||document.documentElement.clientWidth || document.body.clientWidth||0;

clientHeight:window.clientHeight||document.documentElement.clientHeight || document.body.clientHeight||0

} 利用对象的方法进行传递

}

1. 拖拉事件-onresize
2. window.onresize=function(){

Console.log（document.body.clientwidth）

}

**当页面尺寸发生变化就调用,主要是为了取可视区域的宽和高**

1. **响应式布局，**根据设备的大小，调整内容。
2. 事件对象-event：每次触发事件时，系统自动创建一个对象，包含了一些事件触发时相关信息；
3. 获取事件对象，可以在函数括号中写个参数（e/event），打印出来就可以，但是这种方法在IE678，里面不兼容；

兼容写法： document.onclick=function（e）{

e=event||window.event

Var pageY=event.pageY||scroll().top+event.clientY;

Var pageY=event.pageX||scroll().Left+event.clientX

页面位置的兼容和事件对象兼容一起

}

2、事件中的三大坐标

1. screenX/screenY 相对于屏幕左上角的水平和垂直位置；
2. pageX/pageY 相对于页面（body）的水平和垂直位置，要做兼容；
3. clientX/clientY 相对于可视区域左上角的水平和垂直位置；

# webAPI 第八天

1. 事件监听
   * 1. btn.onclick=function{}，类似这些绑定方法容易被层叠，所以有了事件监听器（**addEventListener**）;
     2. **addEventListener,有三个参数，一个是元素，一个是方法，不加on，三是函数；IE8之前有兼容问题,**

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

btn.addEventListener(“click”,fn（匿名函数）)；

* + 1. **attachEvent适用于IE8之前的事件监听，有连个参数，一个是事件，必须要加on，一个是函数，用法和上面相同；**

btn.attachListener(“onclick”,function{})或

btn.attachListener(“onclick”,fn)

* + 1. 兼容写法

Function addevent(ele,eventName,f){

if(ele.addEventListener){

ele.addEventListener(“eventName”,”f”)

}else(ele.attachEvent){

ele.attachEvent(“on”+eventName,”f”)

}

}

* + 1. 删除事件，removeEventListener（eventName,f），

detachEvent(“on”+eventName,f)。匿名函数没办法删除。

Function removeEvent(ele,eventName,f){

if(ele.removeEventListener){

ele.removeEventListener(“eventName”,”f”)

}else(ele.attachEvent){

ele.detachEvent(“on”+eventName,”f”)

}

}

# JS高级第一天

## 数据检测类型

1.1 type of（需要检测的内容）得到的是string类型；

1.2 Xxx instance of 不够严谨，且得到的是布尔类型；

1.3 object.prototype.to String.call（需要检测的内容），得到string类型，这种方法应用较多

## in 的用法

1. 判断在某个对象中能否访问某个属性/方法：语法：”属性名”in 对象名；有就返回true，没有返回false；
2. 判断数组中有无对应的下标；

## Delete的用法

1. 可以删除不规则变量；
2. 删除对象中的属性和方法，但不能删除dom中的属性和方法；
3. 也可删除数组中的元素，原元素变为undefined；

## 面向对象：是面向过程的一种封装

4.1 面向对象：注重结果；

4.2 面向过程：注重过程；

**5、原型：**创建任意一个构造函数，系统都会自动创建一个对象，就是原型。

1. 构造函数中存放方法，有占用/浪费内存的缺点，把匿名函数提炼出来，让发发只想提炼出来的函数，但会出现变量污染的问题--用对象方法进行封装；
2. 原型是对象，可以往中添加属性和方法，存放的是公共的数据。
3. 如何访问原型：构造函数.prototype
4. 添加属性和方法：构造函数.prototype.属性名=
5. 需要注意的点：对象访问成员，先看自己有没有，有的话用自己的成员，否则采用原型中的；修改原型，创建一个对象，要看创建是在修改前还是在修改后；

6、-proto-属性：是对象的属性，指向创造这个对象的构造函数对应的原型：对象.-proto-。开发环境中不用，自己调试时使用，因为其有兼容性。

7、Constructor属性：原型对应的构造函数。

## **8、原型三角关系图**

.prototype

构造函数.prototype原型

构造函数

prototype

.constructor

实例化 .\_\_proto\_\_访问成员

对象

1. **JavaScript 的复合类型：**

Array Math Date RegExp String Number Boolean 还有两个核心类型: Object Function

# JS高级第二天

1. **面向对象的三大特征**
2. **封装：功能封装，整合到对象中，外界使用时只需知道怎样使用这个接口就可以；**
3. **继承：在JS中，继承是对象之间的，一个对象想拥有另一个对象的成员，就让这个继承自另一个对象，其他语言中，继承是类与类之间的；**
4. **多态；**

**2、JS中没有多态这个特征，**他是**基于对象**的语言，也是**轻量型/解释型**语言，在**浏览器**中运行

**3、继承的实现方式**

1. 混入式-利用遍历对象，将对象中的属性赋值给另一个对象

For（var key in JianLin）{

Sicong[key]=JianLin[key]

}

b. 替换原型的方式，将构造函数的原型赋值为想要的对象

Zhanan.prototype=JianLin

c. 混合式

For（var key in JianLin）{

Zhanan.prototype[key]=JianLin[key]

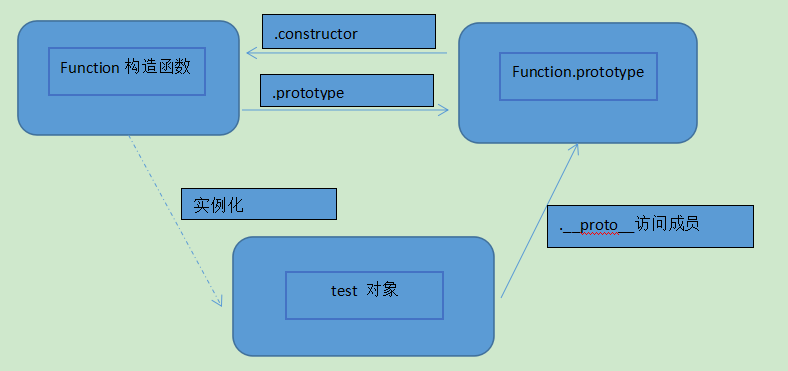
}

1. **原型链**
2. 概念：每一个对象都有一个原型，原型本质上是对象，原型中又有原型，形成一条链就是原型链；最后一个原型为null；
3. 对象和原型中均没有属性，打印为undefined，
4. 对象和原型中没有的方法，打印报错；但是如果原型的原型中有这个方法，就不会报错，如对象.toString();
5. 对象访问成员的规则：自己有则用，没有就沿着原型链找，找到就用，一直也没找到就显示undefined/报错；
6. **Object.prototype成员介绍：**
7. 对象名.hasOwnProperty(“属性名”)：判断对象自己是否有某个属性/方法，返回值是布尔类型，有为true，没有为false；
8. 对象名.isPrototypeOf（对象名）：判断一个对象是否为另一个对象的原型；
9. 对象名.propertyIsEnumerable：判断一个属性是否可以遍历，以及是否属于自己；
10. valueOf:返回原来的值；
11. **怎样安全的给内置对象添加方法**

不建议修改内置对象的构造函数对应的原型，创建一个自己的函数，修改他的原型为空数组，继承数组的属性和方法，再通过原型添加方法和属性。

1. **函数，object也是一个对象**

**函数是由Function（）构造函数创建（实例化出来的）**



7.1：函数成员caller 你这个函数是在那个函数里面被调用的，那么这个caller就是那个函数。

7.2:length,对象的length为形参的个数

7.3：test函数作为一个对象， 点出来的arguments是一个对象，和在函数内部直接输出的arguments不是同一个对象。 我们用的多的是 函数内部直接输出的那个arguments ，有以下两个成员：callee(就这个函数本身）和length（实参的个数）

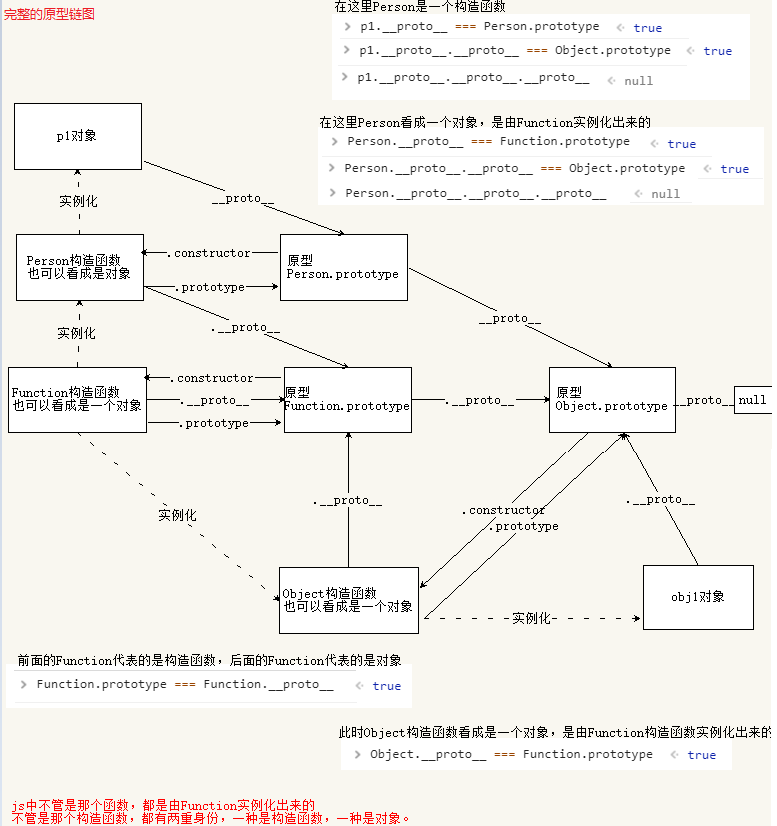
8、Object是一个函数也是一个对象，也是由**Function（）构造函数创建（实例化出来的）**

**9、静态成员和实例成员**

**静态成员：由构建函数点出来的成员；**

**实例成员：由实例化对象点出来的成员；**

**10、完整的原型链图**



# 15、JS高级第三天

1. 递归：函数自己调用自己，需要有个条件让它结束；
2. 闭包：**就是声明在一个函数中的函数**；
3. **作用**：a.延长变量的生命周期；

b.提供有限的访问权限；

1. **使用场景--沙箱**
2. 沙箱是与外界隔绝的一个环境，外界无法修改该环境中的任何变量；
3. 沙箱将所有的变量声明放在最上方，逻辑代码放在中间，如果需要，通过window暴露一些成员；但不可以直接引用，这样会破坏沙箱的结构，应该使用传参的方式将window传进去；
4. 通过自调用函数（IIFE）实现，因为自调用函数不会在外界暴露任何变量，担忧可以形成一个封闭的空间；

自调用函数：（function(){ 代码体

}()）

1. 上下文调用模式
2. call() ,语法：函数名.call(this需要指向的对象，参数1，参数2)；

还有一个用法是，借用object原型中的to string方法来检测数据类型，返回的是该对象类型[ object xxx ]的字符串；

1. apply（）,语法：函数名.apply(this需要指向的对象，数组/伪数组），apply会将数组/伪数组中的元素一个个的提取出来；
2. band(),返回的是一个新的函数，改变了this指向的新函数；
3. 注意：如果第一个参数没写，或者参数为null/undefined/window，不管是谁调用，this都指向window；
4. 伪数组转为数组
5. 设置一个新数组，将伪数组的值遍历出来放入新数组中，得到一个新数组；
6. 设置一个新数组，通过apply方法将伪数组中的数push到新数组中，arr.push.apply(arr ,dirArr);
7. 设置一个新数组，利用pply方法将伪数组中的数concat到新数组中,此时需要接收返回值 arr=arr.concat.apply(arr ,dirArr);
8. 借用数组原型的slice方法，改变this的指向来调用 Array.prototype.slice.call(dirArr,0)

案例：1、求1-n的阶乘，用递归；

2、求1-n的和，用递归；

3、用递归求斐波那契数列，并解决斐波那契数列的性能问题；

4、沙箱