# 一、概述：

## 1、JS是干什么的

Js定义了网页的行为

## 2、js历史

1995年5月 ，网景公司，LiveScript

1995年12月 ，改名为JavaScript

1995年8月 ，微软出了JScript

1997-1999，ECMA国际组织，ECMAScript规范，JavaScript和JScript都要遵守此规范

## 3、特点：

JavaScript是脚本语言，无需编译，(与Java不一样)

JavaScript基于对象，弱类型语言

Html不区分大小写，JS区分

JS语句末尾可有可无分号

## 4、优点

交互性：可以与用户进行动态交互

安全性：js只能与浏览器内运行，不能访问本地硬盘或其它资源

跨平台：只要有浏览器即可，与平台无关

# 二、在HTML中引入JavaScript

## 方式一：HTML中书写 head中或者body中

<script type="text/javascript">

alert("提示信息")

</script>

## 方式二：引入外部js，head中或body中

<script type="text/javascript" src=" application.js"> </script>

# 三、JS语法

## 1、注释：

单行注释：//注释内容

多行注释：/\* 注释内容 \*/

## 2、数据类型：

**①、基本数据类型：**

**A、数值类型(number)**

Js所有数值底层都是浮点型，浮点型和整型之间会自动转换

如：2.4+3.6=6

特殊值：Infinity 正无穷、-Infinity 负无穷

NaN (Not a Number)非数字

IsNaN(参数)，参数数字 返回false，非数字返回true

NaN与任何操作数进行关系比较,结果都是false

**B、字符串类型(string)：**

var x=’ 字符串’ var x=”字符串”

**C、布尔类型(boolean)**

false**和**true

**D、undefined类型**

undefined，变量已声明，未定义(赋值？？)

表示变量未定义，如声明了一个变量，没有初始化，这个变量就是undefined

如：var a; 此时a的值为undefined

**E、null类型**

null 表示空，一般作为返回值使用 ，”” 和 ‘ ’为空字符串

**②、复杂数据类型**

数组、对象都是复杂数据类型

例：数组：var arr=[111,"AAA",true]

## 3、变量的定义 var指向任何类型

var i=’123’ var i=”123”

## 4、运算符 (和java大致相同)

不同之处：

**A、Js中有==和===**

==底层会转为相同数据类型，再比较

===会先比较数据类型,不一样直接返回false否则 比较值

**B、typeof**

返回变量的数据类型 例：typeof v ='string'

**C、delete**

用来删除数组的元素或对象的属性等

var arr=[111,"AAA",true]

delete arr[1] [111,,true] 原来的位置还在

alert(arr[1]) undefined

## 5、语句

for do…while while if …else

(和java基本一致，但js没有增强for循环)

## 6、函数

**A、定义**

**function** 函数名(形参列表){ //形参可以不用写var 写了会报错.

函数体

或**return** 或 **return** 值 可写可不写

}

没有重载

**B、调用**

函数名(实参列表)

## 7、数组

**A、创建数组**

**var** arr1=**new** **Array**(10)//创建10个长度的空数组

**var** arr2=[“abc”,18,true]

**B、注意**

①、可以存放任意类型元素

②、如果某个位置没有元素，该位置的值为**undefined**

③、长度可变 ：arr2.**length**=5

var arr1 = ["123", 411, "55"]

arr1.length = 2

alert(arr1)//123,411,多余元素会删除6

## 8、API 参见菜鸟教程

#### （一）、String 对象

①创建 var str=new String(“abc”)

var str=”abc”

②常用属性和方法

var regexp = /正则内容/ 定义正则

属性length 字符串长度

**A方法str.mactch(regexp)**

var str = "java xx java 哈哈哈"

var regex = /java/g //g表示全部全局 global

alert(str.match(regex) //java,java

**B 方法 str.replace(regexp,replacetext)**

var str = "java xx java 哈哈哈"

var regex = /java/ig //i表示ignorecase 不区分大小写

alert(str.replace(regex, "a")) // a xx a 哈哈哈

**C 方法str.search(regexp) 第一次匹配到的正则位置**

#### （二）、RegExp 对象 (正则)

**1、创建：**

var reg=new RegExp("","")

var reg=/java/[ig]

i:ignorecase 不区分大小写

g :global 全局全部

^必须以什么开头 /^xxx/ 必须以xxx开头

$必须以什么结尾 /yyy$/ 必须以yyy结尾

/^xxx$/ 全字匹配 只能字符串”xxx” 能true 只能包含这个xxx

“xxx java xxx”也为false

**2、使用**

reg.test(str) 用reg正则匹配str 匹配返回true

#### （三）、Array 对象

var arr = new Array(10)

arr.sort()//字典排序

#### （四）、Math 对象

Math可以直接使用，无需创建

Math.ceil(3.14)        向上取整

Math.floor(3.14) 向下取整

Math.round(3.14) 四舍五入

Math.random() 返回0-1之间的随机数

没有构造函数，使用类似Java的Math类

获取[1,100]随机数

Math.ceil(Math.random \* 100) //ceil向上取整0.几都为1

获取 [30,50]随机数

Math.round(Math.random() \* 20 + 30)

#### （五）、全局 对象

直接调用方法即可，比如全局调用isNaN判断是否为非数字

重要: eval 将字符串当作代码来执行

例：eval (“alert(123)”)

#### （六）、自定义对象

**A:定义**

var person={

"name":"张山",

"age":18,

"friends":["a","b","c"] ,

"method":function(){

alert("HH")

}

}

符合json格式，最后一个属性不能有逗号

**B:取值**

person.name 如果name为变量，只能用person[name]

或 person["name"]

**C:调用方法**

person.方法名()

例:person.method()

# 四、DHTML

## 概述

* + 1. 将现有的HTML、css、JS整合在一起，形成了dhtml技术

## 分类

* 1. **BOM：浏览器对象模型，封装了浏览器相关操作**
  2. **DOM：文档对象模型，将Html文档按照文档结构形成树形结构**

## BOM

### **3.1 Window对象**

* + 1. **调用此方法时，可以省略window 比如alert**
    2. **常用事件**
       1. onclik 点击
       2. onfocus 获得焦点
       3. onblur 失去焦点
       4. onload 文档加载完成后触发
       5. onchange 切换/变化

例如：onload=function(){

文档加载完成触发

}

因为文档从上往下加载，如果js在上面，想要操作文档元素会报错，因为文档元素没有加载完成。

如果想要写在上面只能通过事件触发

* + 1. **常用方法**
       1. alert() 弹出消息框
       2. confirm(“确认吗”)弹出确认框
          1. 确定 返回true
          2. 取消 返回false

## DOM

### 获取元素

* + 1. document.getElementById(“id值”); 获取一个元素
    2. document.getElementsByName(“name值”) 获取多个元素(name值可重复)，返回数组
    3. document.getElementsByTagName(“标签名”) 比如input
       1. document.getElementsByTagName(“input”)
    4. 获取值
       1. dc.value 获取值
       2. dc.value=值 设置值
       3. dc.innerText 有些浏览器不支持获取或者设置元素的文本内容

例：

<p id=”x”><input>hello</input></p>

document.getElementById(“x”).innerText //hello

* + - 1. dc.innerHTML 获取该元素下的HTML如果有子元素会得到含标签的信息

例：

<p id=”x”><input>hello</input></p>

document.getElementById(“x”).innerHTML

// 结果 :<input>hello</input>

### 4.2元素的增删改

* + 1. document.createElement(“div”) 创建一个div元素
    2. 元素.removeChild(子元素) 移除子元素
    3. 元素.appendChild(子元素) 末尾增加子元素
    4. 元素.replaceChild(新子元素,旧子元素) 新替换旧
    5. 元素.insertBefore(新元素,旧元素) 在旧元素前面插入新元素
    6. 元素.cloneNode([boolean]) 克隆元素返回，为true克隆包括子
    7. 元素.parentNode.getElementsByTagName("div") 获取当前元素的父节点下的div标签