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

1　概述

1.1.1　将现有的 HTML、css、JS 整合在一起，形成了 dhtml 技术

2　分类

2.1　**BOM：浏览器对象模型，封装了浏览器相关操作**

2.2　**DOM：文档对象模型，将 Html 文档按照文档结构形成树形结构**

3 BOM

3.1 Window 对象

3.1.1 调用此方法时，可以省略 window 比如 alert
3.1.2 常用事件
3.1.2.1 *onclik* *点击*
3.1.2.2 *onfocus* *获得焦点*
3.1.2.3 *onblur* *失去焦点*
3.1.2.4 *onload* *文档加载完成后触发*
3.1.2.5 *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
3.1.3 常用方法
3.1.3.1 alert() 弹出消息框
3.1.3.2 confirm(“确认吗”)弹出确认框
3.1.3.2.1 确定 返回 true
3.1.3.2.2 取消 返回 false

4 DOM

4.1 获取元素

4.1.1 `document.getElementById(“id 值”)`; 获取一个元素
4.1.2 `document.getElementsByName(“name 值”)` 获取多个元素(name 值可重复)，返回数组
4.1.3 `document.getElementsByTagName(“标签名”)` 比如 `input`
4.1.3.1 `document.getElementsByTagName(“input”)`
4.1.4 获取值
4.1.4.1 dc.value 获取值
4.1.4.2 dc.value=值 设置值
4.1.4.3 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
```

4.1.4.4 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

4.1.5 `document.createElement(“div”)` 创建一个 div 元素

4.1.6 元素.`removeChild`(子元素) *移除子元素*

4.1.7 元素.`appendChild`(子元素) *末尾增加子元素*

4.1.8 元素.`replaceChild`(新子元素,旧子元素) *新替换旧*

4.1.9 元素.`insertBefore`(新元素,旧元素) *在旧元素前面插入新元素*

4.1.10 元素.`cloneNode`([boolean]) *克隆元素返回，为 true 克隆包括子*

4.1.11 元素.`parentNode.getElementsByTagName`("div") *获取当前元素的父节点下的 div 标签*