目录

[JavaScript 3](#_Toc118714570)

[简述与问题 3](#_Toc118714571)

[快速入门 3](#_Toc118714572)

[语法入门 4](#_Toc118714573)

[URL\*\*\* 5](#_Toc118714574)

[数据类型 6](#_Toc118714575)

[数据类型检查 6](#_Toc118714576)

[数值型数据\* 6](#_Toc118714577)

[字符串\*\*\* 7](#_Toc118714578)

[布尔值\* 8](#_Toc118714579)

[null和undefined 8](#_Toc118714580)

[数据类型转换 8](#_Toc118714581)

[其他数据类型转化成布尔值 9](#_Toc118714582)

[逻辑运算 9](#_Toc118714583)

[浮点数问题 9](#_Toc118714584)

[数组 10](#_Toc118714585)

[对象 10](#_Toc118714586)

[数据类型总结 11](#_Toc118714587)

[流程控制 12](#_Toc118714588)

[严格检查模式 12](#_Toc118714589)

[流程控制 13](#_Toc118714590)

[Map和Set 14](#_Toc118714591)

[Iterator遍历map和set 14](#_Toc118714592)

[函数 15](#_Toc118714593)

[函数参数问题 16](#_Toc118714594)

[函数的return 17](#_Toc118714595)

[this指向 17](#_Toc118714596)

[变量的作用域 17](#_Toc118714597)

[方法 19](#_Toc118714598)

[JS定时器 20](#_Toc118714599)

[JS动画函数 20](#_Toc118714600)

[获取元素样式 21](#_Toc118714601)

[JS内置对象 21](#_Toc118714602)

[date对象 21](#_Toc118714603)

[JSON字符串 22](#_Toc118714604)

[面向对象编程 23](#_Toc118714605)

[垃圾回收机制 25](#_Toc118714606)

[操作BOM对象\*\*\* 25](#_Toc118714607)

[JS和浏览器简介 25](#_Toc118714608)

[浏览器内置JS对象 26](#_Toc118714609)

[JS事件 28](#_Toc118714610)

[操作DOM对象\*\*\* 28](#_Toc118714611)

[获得节点 29](#_Toc118714612)

[创建新的节点 30](#_Toc118714613)

[创建节点 31](#_Toc118714614)

[插入节点 31](#_Toc118714615)

[更新节点\*\*\* 31](#_Toc118714616)

[删除节点 31](#_Toc118714617)

[操作表单 32](#_Toc118714618)

[表单验证，加密，提交 32](#_Toc118714619)

[JS正则表达式 33](#_Toc118714620)

[JS常用函数 34](#_Toc118714621)

[jQuery 35](#_Toc118714622)

[jQuery应用公式 35](#_Toc118714623)

[选择器selector 36](#_Toc118714624)

[事件action 36](#_Toc118714625)

[jQuery操作DOM 36](#_Toc118714626)

[AJAX和JSON 37](#_Toc118714627)

[AJAX 37](#_Toc118714628)

[仿前端页面小技巧 39](#_Toc118714629)

# JavaScript

## 简述与问题

1. JavaScript是一款最流行的脚本语言。
2. 一个合格的后端人员必须精通Javascript，因为JS可以操控DOM和BOM
3. ECMAScript可以理解为是JavaScript的一个标准，最新版本已经到了es6,但是大部分浏览器还只停留在支持es5代码上！
4. 开发环境和线上环境不一致

问题：切忌，script标签不能自闭和，必须成对出现！

## 快速入门

内部标签

<!-- <script>-->  
<!-- alert('hello world')-->  
<!-- </script>-->  
 <!--切忌：script标签必须成对出现，否则不会生效，不需要显示定义type,默认就是javascript-->

外部引入

<script src="Test01.js"></script>

完整实例如下

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Title</title>  
<!-- <script>-->  
<!-- alert('hello world')-->  
<!-- </script>-->  
 <!--切忌：script标签必须成对出现，否则不会生效，不需要显示定义type,默认就是javascript-->  
 <script src="Test01.js"></script>  
</head>  
<body>  
 <!--这里也可以存放-->  
</body>  
</html>

## 语法入门

<script>  
 // 变量的定义 关键字var  
 var ***a*** = 1;  
 alert(***a***);  
 // 条件控制  
 if(2>1){  
 if(3>2){  
 alert("hello");  
 }  
 alert(true);  
 }else{  
 alert("mf");  
 }  
 //console.log(a)在浏览器的控制台打印变量  
</script>

## URL\*\*\*

URL全称：uniform resource location统一资源定位符

URL的构成：

* 协议scheme：通信协议，常用的http,https
* 主机名host：IP地址或者主机名，用://与协议分开
* 端口port：端口号，用:与主机IP分开
* 路径path：路径，主机上的一个文件资源路径
* 查询query：查询，用于给动态网页传递参数！用？与路径分开表示参数，参数与参数之间用&分开！参数就是键值对key=value!
* 锚点fragmeng：网页加载完成后滚动到锚点位置！用#与查询分开，#之后就是锚点！

## 数据类型

数值，文本，图形，音频，视频

注意：变量的声明不能以数字开头

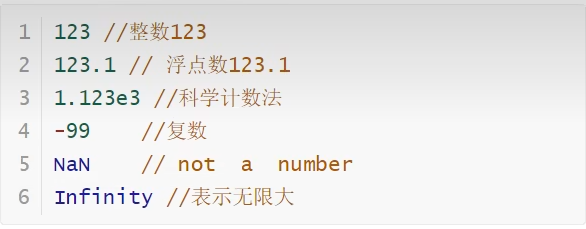
### 数据类型检查

typeof关键字检查一个变量的数据类型，用法：

typeof 变量

### 数值型数据\*

数值型数据在js不区分小数和整数，且不可修改，只要两个值相等，我们就认为相等



### 字符串\*\*\*

‘abc’ 和 “abc”

1. 使用\给特殊字符转义

***console***.log('a');  
***console***.log("b");  
***console***.log("a\'\nb");

1. 字符串拼接

let ***name*** = "孙道名";  
let ***msg*** = "你好";  
let ***xx*** = ***name***+***msg***;

字符串长度str.length

js字符串和java字符串一样，都是不可变的

js字符串大小写转化方法，toUpperCase()和toLowerCase()

subString(a,b):将字符串从a开始截取到b

indexof(),replace()等待

在JS中提供了三个包装类，通过这三个包装类可以完成基本数据类型转化成对象

* String(x)：基本数据类型x转化为String对象
* Number(x)：x数值型转化为Number对象
* Boolean(x)：布尔值x转化为Boolean()对象

字符串常用方法：

* + charAt(index)：定位index位置上的字符
  + concat(s1,s2,s3…)：连接字符串s1,s2,s3……,参数任意个数
  + indexof(“s”)：返回字符串s在主串中第一次出现的位置索引
  + lastindexof(“s”)：返回字符串s在主串中最后一次出现的位置索引
  + subString(“index”)：从index位置开始截取

### 布尔值\*

和java一样

### null和undefined

空和未定义

### 数据类型转换

* 数值转换成字符串：
  + 调用无参方法toString()，空和未定义数据类型没有该方法

var ***a*** = 100;  
var ***b*** = ***a***.toString();  
alert(***b***);  
alert(typeof ***b***)

* + 调用String(target)方法，并且把要转类型的数据作为String方法的参数
  + 拼串：数据+空串完成字符串的转化！
* 字符串转化成数值型
  + Number()
    - 如果是纯数字字符串，则转化成数字。
    - 如果字符串带有非数字内容，则转化成NaN
    - 如果字符串是一个空串，或者是全空格字符串，则转化成0
    - 布尔转数字，true为1，false为0
  + parseInt(s)
    - 把字符串s转化成整数
  + parseFloat(s)
    - 把字符串s转化成浮点数

### 其他数据类型转化成布尔值

使用Boolean（）函数

* 数字处了0和nan，都是true
* 字符串除了空串都是true
* 对象为true

### 逻辑运算

&&：逻辑与

||：逻辑或

！：非

运算符号 js和java区别较大的点

=：赋值

==：等于，类型不一样，值一样，也会返回true(坚持不要使用==，使用绝对等于)

===:绝对等于，类型和值都一样，才返回true

NaN===NaN:这个与所有的数值都不相等，包括自己

只能通过isNaN(NaN)来判断这个数是否是NaN

### 浮点数问题

1/3 和 1-2/3是不等的。因为1/3是无限循环浮点数，在计算机中会被净化成有限个数，所以二者不等

尽量避免使用浮点数数进行运算，存在精度问题，如果需要对浮点数进行运算，为避免精度问题影响结果，可以使用Math.abs（和java一样）

### 数组

var arr = [1,”ab”,’c’,null,true];

new Array(1,”ab”,’c’,null,true);

因为js数据不分类型，所以数组可以存在任意数据类型

如果访问数组越界，则会报未定义undefined

如果给数组的长度赋值就会使数组扩容减容。扩容的空间默认是空。减容削减的空间数据自动丢失

indexof(目标)方法：获取目标索引

slice()方法：截取数组的一部分返回一个新的数组，类似于String的substring

push和pop方法：数据压入数组尾部方法和弹出数组尾部数据方法。针对于数组的尾部

unshift和shift方法:………………………………………………………………………针对于数组的头部

sort方法排序

reverse方法元素反转

数组的拼接，concat方法和join方法，前者返回一个新的数据，原数组不变，join拼接数组

多维数组

数组（存储数据）：如何存，如何取

### 对象

对象的创建方式有两种，如下：

let ***person*** = new Object();  
***person*** = {  
 name:"asm",  
 age:99,  
 sex:'male'  
}  
  
let ***me*** = {  
 name:"sdm",  
 sex:'male',  
 data:'more'  
}

本质上是若干键值对

对象是大括号，数组是中括号，对象的定义和基本数据类型一样用var关键字

对象属性用键值对表示，不同属性之间用逗号隔开，最后一个属性不用逗号



取值和赋值：对象.属性



删除对象的属性：delete 对象.属性

添加对象的属性：对象.新属性 = 值

js中所有键都是字符串，值可以是任意对象

判断属性是否在这个对象中

‘属性’in 对象 ：返回布尔值

和java类似，js对属性和方法的调用区别是一个括号！

### 数据类型总结

Javascripte数据类型总体分为两类，和java一样，基本数据类型和引用数据类型。

基本数据类型：数值型，字符串，null，undefine,布尔

数值型数据的值相等就是相等的，不分整数浮点数等等

引用数据类型的值是保存在内存中的对象，当一个变量是一个对象时，这个变量保存的是这个对象的引用。和JAVA类似，内存中的堆栈内存，堆保存对象，栈存放对象的引用和基本数据类型。

## 流程控制

### 严格检查模式

‘use strict’:严格检查模式，预防js的随意性产生的问题,前提是idea支持es6语法，必须js代码中的第一行

创建局部变量推荐使用let，而不是var

使用如下：



### 流程控制

if else if else 语句参照java

while循环参照java

do while循环参照java

for循环参照java,局部变量定义用let

foreach循环for in把java中的分号改为in,类似于python

let ***array*** = [1,2,'3','4','a',true];  
  
for(let i in ***array***){  
 ***console***.log(***array***[i]);  
}  
***array***.forEach(function (item) {  
 ***console***.log(item)  
})

### Map和Set

map

let ***map*** = new ***Map***([['tom',90],['john',80],['tom',70]]);  
let ***score*** = ***map***.get('tom');//取值  
***map***.set("admin",199);//添加值  
***console***.log(***score***);

map的遍历

let ***map*** = new ***Map***([['tom',90],['john',80],['tom',70]]);  
//for of 遍历map,把每一个entry以数组形式输出  
for(let i of ***map***){  
 ***console***.log(i);  
}  
//forEach遍历map,以两个变量形式输出key,val  
***map***.forEach(function (key,val){  
 ***console***.log(val);  
 ***console***.log(key);  
})

Set:无序不可重复集合，可以去重

//重复的元素会被自动省略掉  
let ***set*** = new ***Set***([2,1,1,2]);

set.add(3);//增加元素

set.delete(1);//删除元素

set.has(value);//set是否包含某个元素，返回布尔值  
***console***.log(***set***);

### Iterator遍历map和set

利用for语句遍历数组，set和map

let ***array*** = [1,'a',"abc",true]  
for(let i in ***array***){  
 ***console***.log(***array***[i]);  
}  
let ***map*** = new ***Map***([['a',99],[true,'100'],['xxx',90]])  
***map***.forEach(function (key,value) {  
 ***console***.log(key);  
 ***console***.log(value);  
})  
let ***set*** = new ***Set***(['a','bcd',12,false]);  
for(let i of ***set***){  
 ***console***.log(i);  
}

## 函数

函数的定义1:function 函数名（参数名）{}，参数不需要使用var或者let定义

function abs(x){  
 if(x>=0){  
 return x;  
 }else{  
 return -x;  
 }  
}

执行return语句，返回结果，函数结束

没有执行return，函数执行完也会返回结果，结果就是undefined

函数的定义2：var abs = function(x){} 把函数赋值给abs,通过abs调用函数

let bbs = function (i){  
 return i>=0?i:-i;  
}  
***console***.log(bbs(-100))

### 函数参数问题

js可以传递任意个数参数，也可以不传参数！对于函数而言只接受对应数量的前几个参数

规避函数无参数问题，对函数参数进行判断抛出异常检查

if(typeof i!=='number'){//===表示等于，不等！==  
 throw 'you fking idiot';  
}

arguments是js中的关键字，利用arguments获取传递进去的参数

function abs(x){  
 for(let i = 0;i<arguments.length;i++){  
 ***console***.log(arguments[i]);  
 }  
 if(x>=0){  
 return x;  
 }else{  
 return -x;  
 }  
}

传统方法中，获取函数的超额所有参数：

function abs(x){  
 if(arguments.length>1){  
 for(var i = 1;i<arguments.length;i++){  
 ***console***.log(arguments[i]);  
 }  
 }  
 if(x>=0){  
 return x;  
 }else{  
 return -x;  
 }  
}

新版本ES6提供的方法中：参数...reset,超额参数会以数组形式返回，且只能写在参数最后面

function abs(x,...reset){  
 ***console***.log(reset)  
 if(x>=0){  
 return x;  
 }else{  
 return -x;  
 }  
}

### 函数的return

return true返回正常或者默认结果

return false阻止默认结果或者终止处理

### this指向

全局作用下，this指向window

函数内的this指向调用函数的那个对象

### 变量的作用域

#### 基本作用域

var定义的变量有作用域，如果在函数体中声明，则函数体外不可以使用，反之，定义在外部的变量，其下的函数可以使用这个变量

function test(){  
 var x;  
}  
x = 10;//x在严格检查模式下报错

存在嵌套函数，对于var定义的变量，内部函数可以访问外部变量，外部不能访问内部成员

function test(){  
 var x;  
 function testson(){  
 x = 10;  
 var y;  
 }  
 y = 10;//这里y报错  
}

javascript函数查找变量从自身函数开始，由内向外查找，假设外部存在这个同名的函数变量，则内部函数会屏蔽外部函数的变量。

#### ES6规范和作用域

js中会自动声明变量，但是不会赋值，如下代码中输出的是xundefined

function test(){  
 var x = 'x' + y;  
 var y = 'y';  
 ***console***.log(x);  
}  
test();

这个是在JS建立之初就存在的特性，所有定义的变量写在函数的头部

alert()函数本身也是一个window对象，JS实际上只有一个全局作用域window，任何变量(函数)没有在函数作用范围内找到，就会向外查找！如果在全局作用域都没有找到，就会报引用异常。

规范：由于我们所有的全局变量都会绑定到我们的window中，如果不同的JS文件使用了相同的全局变量，如何降低冲突？

定义一个独一无二的全局变量，定义其他变量时使用这个唯一的全局变量来定义！本质上是利用JS的对象来完成

var ***SDM*** = {};  
***SDM***.alert = function (str){  
 ***window***.alert(str);  
}  
***SDM***.alert("你好");

#### 局部作用域let

for(var ***i*** = 0;***i***<10;***i***++){  
 ***console***.log(***i***);  
}  
***console***.log(***i***+1);//出了作用域i依旧可以使用

为了解决如下问题，使用let关键字声明变量可以避免以上问题，推荐使用let定义局部变量

#### 常量const

在ES6之前，怎么定义常量：只有用全部大写的字母表示常量(约定大于配置)

在ES6引入了常量关键字const,该关键字声明的常量不可变

const ***arr*** = [];  
***arr***[0] = 1;//没有改变常量的地址  
***arr*** = ['aa'];//改变了地址，所以报错

总结

常量用const声明,const声明的常量地址不可变！

### 方法

理解：对象中的函数就是方法，对象包括方法和属性

方法的调用需要带括号，属性则不需要括号

var ***SDM*** = {  
 name:"孙道名",  
 birth:1999,  
 age:function (){  
 let now = new ***Date***().getFullYear();  
 return now - this.birth; }  
};  
***console***.log(***SDM***.age());//调用方法  
***console***.log(***SDM***.name);//调用属性

注意：这里age方法使用了this关键字，指代当前对象。没有this则默认调用系统window中绑定的birth,因为没有所以会报birth没有被定义。

在java中this是无法指向的，默认指向调用它的那个对象。

在js中，this默认指向调用它的那个对象，但是可以改变指向。通过apply指定目标对象

function getAge(){  
 let now = new ***Date***().getFullYear();  
 return now - this.birth;  
}  
var ***SDM*** = {  
 name:"孙道名",  
 birth:1999,  
 age:getAge//方法的拆分写法不需要带括号   
};  
***console***.log(getAge.apply(***SDM***,[]));//this指向SDM,参数为空

### JS定时器

JavaScript有两种定时器：

* setTimeout()：指定时间后执行一段代码，延迟执行
  + 定时器取消：clearTimeout(定时器)
* setInterval(function(){},毫秒数)：每隔一段时间执行一段代码
  + 定时器取消：clearInterval(定时器)

每次使用定时器时需要先清除定时器再使用！

### JS动画函数

animation()

animate()函数

示例：

/\*动画函数\*/  
// function animate(element, target) {  
// // 先清理定时器  
// clearInterval(element.timeId);  
// // 一会要清理定时器(只产生一个定时器)  
// element.timeId = setInterval(function() {  
// // 获取对象当前的位置  
// var current = element.offsetLeft;  
// // 每次移动多少像素  
// var step = 10;  
// // 判断是往正方向走还是往相反方向走  
// step = current < target ? step : -step;  
// // 每次移动后的距离  
// current += step;  
// // 判断当前移动后的位置是否到达目标位置  
// if(Math.abs(target - current) > Math.abs(step)) {  
// element.style.left = current + "px";  
// } else {  
// // 清理定时器  
// clearInterval(element.timeId);  
// element.style.left = target + "px";  
// }  
// }, 20);  
// }

### 获取元素样式

getComputedStyle(param1,param2):该方法获取元素当前样式，该方法隶属于window！可以直接使用！

参数1：要获取样式的目标元素

参数2：传递一个伪元素，一般传null！

该方法会返回一个对象，该对象内封装了当前元素对应的样式，可以使用对象.样式名来读取样式！且是只读，不能修改！

### 函数闭包

## JS内置对象

### date对象

var ***now*** = new ***Date***();  
***now***.getFullYear();//年  
***now***.getMonth();//月份，注意这里是0-11月，对应1-12月  
***now***.getDay();//星期几  
***now***.getDate();//日  
***now***.getHours();//小时  
***now***.getMinutes();//分  
***now***.getSeconds();//秒  
***now***.getTime();//时间戳 全世界统一，从1970年1月1日0:00:00 到现在的毫秒数  
***now***.toLocaleDateString();  
***now***.toISOString();

### JSON字符串

简介：一种轻量级数据交换格式,格式和结构简洁清晰，便于人阅读和编写，同时也利于机器解析，能有效提升网络传输效率。

在JS中一切皆为对象，任何JS支持的类型都可以用JSON字符串来表示。JSON就把对象按照格式制定的字符串。

对象：

Object { name: "孙道名", age: 22, sex: "男", study: (4) […] }

该对象对应的JSON字符串数据：

‘{"name":"孙道名","age":22,"sex":"男","study":["java","js","SSM","springBoot"]}’

JSON格式：

* 对象都用{}
* 数组都用[]
* 所有键值对格式都使用key:value

var ***SDM*** = {  
 name:'孙道名',  
 age:22,  
 sex:'男',  
 study:['java','js','SSM','springBoot']  
}  
//对象转化成JSON对象(本质上是字符串)  
let ***sdm*** = ***JSON***.stringify(***SDM***);  
***console***.log(***sdm***);

//把字符串解析成对象  
let ***obj*** = ***JSON***.parse(***sdm***);  
***console***.log(***obj***);

### 面向对象编程

面向对象编程：

在java中

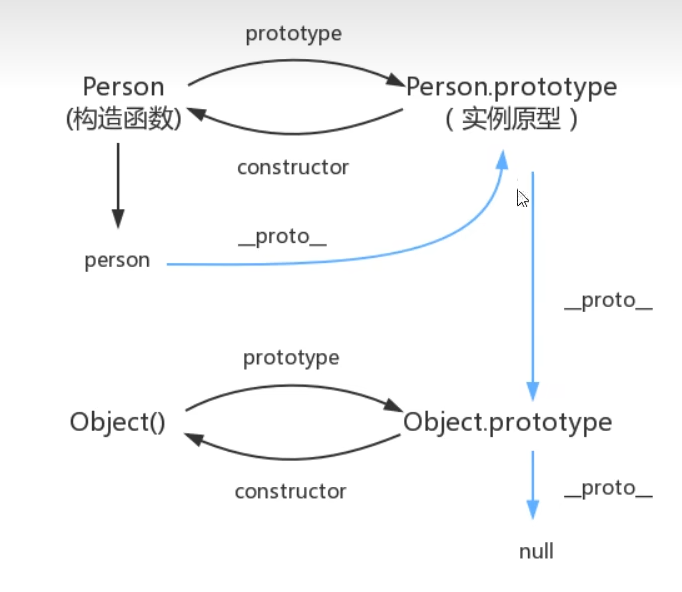
* 类：模板
* 对象：具体的实例

在JS中，继承有两种，一种原型继承，一种class继承，后者更加规范且和java具有一定的共同性，所以推荐使用。

#### 原型继承

var ***People*** = {  
 name:"人",  
 run:function (){  
 ***console***.log(this.name+"run......");  
 }  
};  
var ***Bird*** = {  
 name:"鸟",  
 fly: function (){  
 ***console***.log(this.name+"飞.......");  
 }  
};  
var ***me*** = {  
 name:"ss"  
};

//proto前后都是双下划线  
***me***.\_\_proto\_\_ = ***Bird***;//原型链，将me的原型指向Bird,类似于java的继承，

原型链：如图

在JS中，每个对象都会在其内部初始化一个属性，就是prototype(原型)。

当我们访问一个对象的属性时，如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，于是就这样一直找下去，也就是我们平时所说的原型链的概念。

#### class继承

class继承在ES6引入的，意味着有的浏览器不支持

class Student{  
 constructor(name,age) {  
 this.name = name;  
 this.age = age;  
 }  
 hello(){  
 alert(this.age,this.name);  
 }  
};  
class XiaoStudent extends Student{  
 constructor(name,score) {//属性在构造器中定义  
 super(name);  
 this.score = score;  
 }  
 run(){//方法类比java,没有修饰符和返回值，所以只写方法名  
 alert("running.....");  
 }  
};  
var ***xm*** = new Student("xiaoming",111);  
var ***xiaoxs*** = new XiaoStudent("xiaosues",99);

和java非常相似且规范。推荐只使用此类继承

## 垃圾回收机制

JS中拥有自动垃圾回收机制，对于不需要使用的对象，只需要将对象设置为null即可！来及回收机制会自动销毁！

## 操作BOM对象\*\*\*

### JS和浏览器简介

JS和浏览器的关系：

* JS的诞生就是为了操控浏览器，浏览器会执行JS代码

BOM:Browser object model,浏览器对象模型

* IE浏览器
* Chrome
* firefox(Linux默认浏览器，开发者常用)
* Safari

以上这些都是原生浏览器,即拥有完整独立内核的浏览器

第三方浏览器：基于IE核心技术开发出来衍生产品。如果电脑上没有IE，则这些浏览器无法运行，如下：

* QQ浏览器
* 360浏览器

### 浏览器内置JS对象

#### window

代表浏览器窗口，窗口的基本长宽属性如下：

***window***.innerHeight;  
***window***.innerWidth;  
***window***.outerHeight;  
***window***.outerWidth;

#### Navigator

封装了浏览器的信息，大多数时候，我们不会使用这个对象。因为会被人为修改

***navigator***.appName;  
***navigator***.appVersion;//获取浏览器版本windows还是android等  
***navigator***.appCodeName;

#### screen

代表整个屏幕信息

#### location \*

封装了当前页面的URL信息，很重要

host:"www.baidu.com";  
href:"https://www.baidu.com/"  
protocol:"https:"  
reload:f reload() //刷新网页  
***location***.assign("URL");//跳转到新的地址

location.href = url：页面跳转至url

#### document网页文本内容\*

document代表当前的页面，HTML DOM文档树

实例：

***document***.title = 'name'//设置页面标题

获取文档树节点(选择器)

前端获取cookie

***document***.cookie

劫持cookie原理：

利用弹窗等获取点击事件，引入js代码获取cookie,远程发送到其它服务器上，利用cookie实现免密登录等操作。

如何避免：服务器端设置cookie为httpOnly

#### history\*

代表浏览器的历史记录

网页的前进后退

***history***.back();//后退  
***history***.forward();//前进

## JS事件

键盘事件：

* onkeydown:键盘被按下
* onkeyup:键盘按下后松开
* onmouseover:鼠标移动到目标
* onmouseout:鼠标移出目标

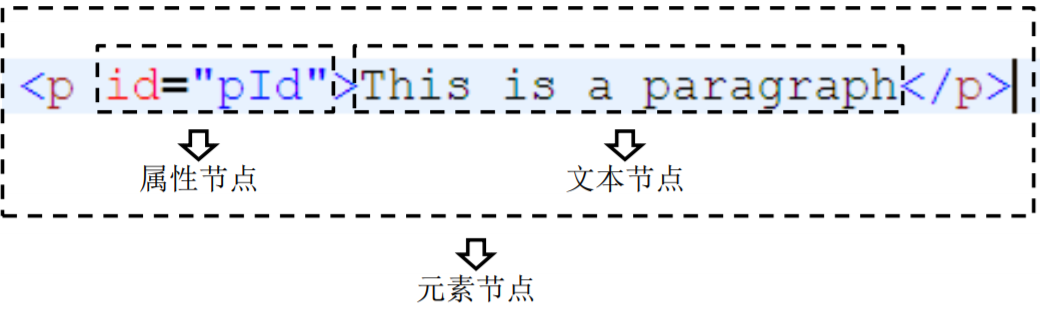
鼠标事件：onclick点击

## 操作DOM对象\*\*\*

DOM（Document Object Model）：文档对象模型

DOM对象的常用节点分为四类：

* 文档节点：整个HTML文档，即document
* 元素节点：整个标签
* 属性节点：标签的属性
* 文本节点：标签内包括的文本



核心：浏览器的网页就是一个DOM树形结构，操作如下：（增删改查）

* 改：更新DOM节点
* 查：遍历DOM节点：得到DOM节点
* 删除:删除一个DOM节点
* 增：添加一个新的DOM节点

### 获得节点

要操作一个DOM节点，就要先得到这个DOM节点，document对象代表了当前整个页面，

可以利用document对象获取目标节点

//这是原生代码，之后尽量都使用Jquery

<div id = "father">  
 <h1>标题1</h1>  
 <p id = "p1">p1</p>  
 <p class="p2">p2</p>  
</div>  
  
<script>  
 //对应CSS选择器  
 var ***title*** = ***document***.getElementsByTagName('h1');  
 var ***p1*** = ***document***.getElementById("p1");  
 var ***p2*** = ***document***.getElementsByClassName('p2');  
 var ***father*** = ***document***.getElementById("father");  
 var ***childrens*** = ***father***.children[index];//根据索引获取父节点下的子节点  
 // father.firstChild;获取第一个最后一个孩子节点等  
 // father.lastChild;  
   
</script>

#### 获取列表节点

对于html中的列表元素li,通过js获取标签li集合！

<body>  
 <ul id="ul">  
 <li>第1个</li>  
 <li>第2个</li>  
 <li>第3个</li>  
 <li id="four">第4个</li>  
 <li>第5个</li>  
 <li>第6个</li>  
 <li>第7个</li>  
 </ul>  
</body>  
  
<script>  
 var ***list*** = ***document***.getElementById("ul").getElementsByTagName("li");  
 alert(***list***.length);//list是所有li标签的集合！

也可以通过li标签的父标签ul的children属性获取！获取目标子标签

var ***x*** = ***document***.getElementById("ul").children;

//获取当前节点的父级节点  
***console***.log(***document***.getElementById("ul").parentNode);

//获取当前节点的第一个子节点  
***console***.log(***document***.getElementById("ul").firstChild); //IE8中是第一个子元素

//获取当前节点的最后一个子节点  
***console***.log(***document***.getElementById("ul").lastChild); //IE8中是第一个子元素

#### 获取input标签的值

通过document获取input标签节点，调用value属性获取值

### 创建新的节点

var ***newp*** = ***document***.createElement("p");//字符串内是标签名  
***newp***.id = "np";  
***newp***.innerText = "易";

//还可以创建JS代码

var ***myScript*** = ***document***.createElement("script");  
***myScript***.setAttribute('type','text/javascript');

//上面两句等效于下面这一句  
<script type="text/javascript" src="">

### 创建节点

var param = document.createElement(“标签名”)

var xx = ***document***.createElement("p");  
xx.innerText = "新的段落";  
***document***.getElementById("h1").append(xx);

### 插入节点

我们获得某个DOM节点，假设这个dom节点是空的，我们通过innerHTML就可以增加一个元素。但是如果这个DOM节点包含了子节点元素，我们增加一个元素就会覆盖原有元素。

所以我们使用append方法

* **节点.append(字符串) :节点末尾添加一段字符串**
* **节点.appendChild(目标节点)：把已经存在的目标节点作为子节点追加进入节点中**

把节点插入到前面

* insert before(new Node,目标节点)：把新节点插入到目标节点前面

### 更新节点\*\*\*

(这两个都是属性)

操作文本内容：注意innerHTML符合W3C标准，innerText只适用于IE浏览器！

* **p1.innerText =“…” ：设置节点内的文本内容**
* **p1.innerHTML =“…”：设置节点内的样式和内容，可以解析标签**

操作CSS样式：注意一下的属性都是根据驼峰命名法

* **p1.style.color = ‘red’**
* **p1.style.fontSize = ‘20px’**
* **p1.style.padding = ‘2em’**

**操作节点属性：在jquery中使用attr()和prop()获取属性，后者通常用于获取布尔值属性！**

**setAttribute(属性名称)**

**getAttribute(属性名称)**

### 删除节点

因为节点的删除不存在自我删除这一逻辑，所以删除步骤如下：

* 利用目标节点获取到他的父节点
* 利用父节点对目标节点进行删除

方法：

* 目标节点.parentElement //获取目标节点的父类节点
* 父节点.removeChild(目标节点) //父节点删除目标节点

注意：删除的过程是动态的，每删除一个节点，其他节点都是在动态变化的

实例：

//利用parentNode这一属性

var target = ***document***.getElementById("h1");  
target.parentNode.removeChild(target);

### 操作表单

表单是什么：

* + 文本框text
  + 下拉框<select>
  + 单选框radio
  + 多选框checkbox
  + 隐藏框hidden
  + 密码框password
  + ……….

表单的目的：提交信息

利用document对象对DOM节点的操控，可以获取表单的数据。

对于单选框，调用它的checked属性（返回布尔值）判断是否为true

### 表单验证，加密，提交

验证：表单的onsubmit属性，绑定js函数进行验证

加密：html文件引入MD5，然后使用MD5算法对目标数据进行加密。

提交：onsubmit属性值是true时，用户点击submit按钮时，提交。为false时，阻止提交。

前端知识补充：

input type = button/submit——》button和submit的区别

input type =button常用于添加onclick事件的监听，如果没有添加onclick事件，点击以后没有任何效果.

input type = submit.默认情况下点击就会提交整个表单数据。

<!-- 下载导入MD5工具类 -->  
 <script src="https://cdn.bootcss.com/blueimp-md5/2.10.0/js/md5.min.js"></script>  
</head>  
<body>  
 <!--表单提交onsubmit需要绑定一个函数，通过return和函数的返回值，完成表单的提交和阻止提交的功能  
 return check()  
 -->  
 <form action="#" method="post" onsubmit="return check()">  
 用户名:<input type="text" name = "username" id="username"><br>  
 密&nbsp;&nbsp;&nbsp;&nbsp;码:<input type="password" id="input-password"><br>  
 <!-- 利用隐藏域 -->  
 <input type="hidden" id="md5-password" name = "password">  
 <!--绑定事件利用onclick属性，onclick事件会在元素被点击时发生-->  
 <input type="submit" value="提交">  
 </form>  
  
 <script>  
 function check(){  
 let uname = ***document***.getElementById("username");  
 let pwd = ***document***.getElementById('input-password');  
 //隐藏域的加密密码  
 let md5pwd = ***document***.getElementById('md5-password');  
 md5pwd.value = md5(pwd.value);  
 ***console***.log(uname.value);  
 ***console***.log(pwd.value);  
 //利用MD5对表单数据进行加密  
 return false;//这里只是演示阻止表单提交  
 };

## JS正则表达式

## JS常用函数

在JS中，可以在指定的时间间隔内执行一段代码，这些事件间隔称为定时事件！它们是：

* + setInterval(函数，毫秒数)
    - 参数1是函数，即要执行的事件函数
    - 参数2是毫秒数整数，即时间间隔！

如果要停止setInterval函数，使用clearInterval(intervalID)方法，intervalID是setInterval方法的返回值！

* + setTimeout()

随机数函数Math.random()

时间函数：内置对象Date

* + 获取时间戳getTime()方法
  + 获取小时getHours()
  + 获取分钟getMinutes()

## jQuery

jQuery，js封装的函数库，！注意，jquery的选择器只是作为选中目标，进而改变目标来使用！而函数的参数亦或者jquery的action参数如果是一个函数则不能使用jquery作为参数！

引入JS

* 下载jQuery源文件引入项目中
* 引入CDN 在线链接

实例代码如下：

//引入CDN 在线链接

<script src="https://code.jquery.com/jquery-2.1.1.min.js"></script>

//本地项目源文件  
<script src="lib/jquery-3.6.1.js"></script>

### jQuery应用公式

$(selector).action()

<!-- 公式：$(选择器).action() -->  
//选择器就是CSS选择器  
$('#test-jquery').click(function (){  
 alert('hello');  
})

### 选择器selector

* 标签选择器
* id选择器
* 类选择器

//jQuery选择器 CSS中的选择器它全都能用  
$('标签名').action();//标签选择器  
$('#id').action();//id选择器  
$('.class').action();//类选择器

### 事件action

详情查询Jquery工具站<https://jquery.cuishifeng.cn/>

* 鼠标事件

$('').mousedown();//按下  
$('').mouseout();//  
$('').mousemove();//移动  
$('').mouseover();  
$('').mouseup();  
$('').mouseenter();  
$('').mouseleave();

键盘事件：详情查询Jquery工具站<https://jquery.cuishifeng.cn/>

其他事件

### jQuery操作DOM

部分实例代码，详情查询Jquery工具站<https://jquery.cuishifeng.cn/>

$('#test-ul li[name=python]').text();//取值  
$('#test-ul li[name=python]').text();//设置值  
$('#test-ul li[name=python]').css('color','red');//css样式操作  
$('#test-ul li[name=python]').show();//元素的显示  
$('#test-ul li[name=python]').hide();//元素的隐藏

## AJAX和JSON

### AJAX

AJAX(Asynchronized JavaScript and XML)，异步javascript 和 xml。

AJAX是一种不需重新加载整个网页的情况下，能够更新部分网页的技术。

传统网页想要更新内容需要重新加载整个网页，使用ajax可以实现局部网页的更新。增强B/S的体验！

B/S架构：未来的主流，并且会爆发式持续增长！

利用JS伪造Ajax,Ajax的底层XMLHttpRequest（xhr请求）

<div>  
 <p>加载地址：</p>  
 <p>  
 <input type="text" id="url">  
 <input type="button" value="提交" onclick="loadpage()">  
 </p>  
</div>  
<div>  
 <h3>加载页面的位置:</h3>  
 <iframe style="width:100%;height:500px" id="iframeposition">  
  
 </iframe>  
</div>  
<script>  
 function loadpage(){  
 var url = ***document***.getElementById("url").value;  
 ***document***.getElementById("iframeposition").src = url;  
 }  
</script>

function al(){  
 */\*\*  
 \* url:待载入页面的url地址  
 \* data:待发送的key/value参数  
 \* success:载入成功时回调函数  
 \* data:封装了服务器返回的数据  
 \* \*status:状态  
 \* ajax默认是get请求  
 \* \*/* $.ajax({  
 url:"${pageContext.request.contextPath}/ajax/a1",  
 data:{"username":$("txtName").val()},  
 success:function (data){  
 ***console***.log(data);  
 ***console***.log(***status***);  
 }  
 });  
  
}

AJAX现有两种，Jquery和Vue,使用哪个就导入对应的文件！

ajax三步：

* 编写对应的请求
* 编写ajax请求
  + url：请求
  + data：键值对
  + success：回调函数
* 给ajax绑定事件

## JS总结

数据结构，流程控制

函数：闭包

DOM：获取节点，创建节点，删除节点

BOM：window,document

## 仿前端页面小技巧

f12打开控制台，查看网页源码，从头开始，delete按键删除网页元素，不需要的不影响网页结构的元素都可以删除，留下前端最核心的代码(如CSS等)。右键复制整个HTML文档。即可模仿该网页。

巩固JS：看Jquery源码，看游戏源码，仿别人前端页面。

或者右键页面另存为，有的网站可以获取整个页面和js,图片资源