JavaScript

1、快速入门

1.1、引入JavaScript

1.1.1、内部标签

```
    <script >
        alert("hello,world")
    </script>
```

1.1.2、外部引用

qj.js

```
<script src="js/qj.js">
</script>
```

1.2、基本语法的引入

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script>
        let score = 66;
        if(score>60 && score<70){
            alert("60-70")
        }else if(score>=70 && score<80){
            alert("70-80")
        }else
            alert("other")
    </script>
</head>
<body>

</body>
</html>
```

1.3、数据类型

数值、文本、图形、音频、视频。。。

number

js不区分小数和整数，number

```
123 //整数123
123.1 //浮点数123.1
-99 //负数
1.123e3 //科学计数法
NaN //not a number
Infinity //表示无限大
```

字符串

'abc' "abc"

布尔值

true、false

逻辑运算

```
&& 两个都为真则为真
|| 一个为真，结果为真
！ 真即假，假即真
```

比较运算符

```
=
==   等于（类型不一样，值一样，也会判断为true）
===  绝对等于（类型一样，值一样，结果为true）
```

须知

- NaN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

尽量避免使用浮点数进行运算，存在精度问题

null和undefined

- null 空
- undefined 未定义

数组

java中必须是一系列相同类型的对象，js中不需要这样

```
<script>
        let a= [1,2,3,"s",null,true]
        new Array(1,2,3,"s",null,true)
</script>
```

如果越界了就会报undefined

对象

对象是大括号，数组是中括号

```
<script>
        let person={
            name : "qinjiang",
            age : 3,
            tags:['js',"java","sk"]
        }
</script>
```

1.4、严格检查模式

'use strict'

2、数据类型

2.1、字符串

1、正常字符串我们使用 单引号，或者双引号包裹

2、注意转义字符 \

3、多行字符串编写

4、模板字符串 ${}

5、字符串长度

6、字符串的可变性，不可变

7、大小写转换

8、student.indexOf('t')

9、substring

2.2、数组

array可以包含任意的数据类型

```
var arr = [1,2,3,1,2,3]
```

1、长度

注意：加入给arr.length赋值，数组大小就会发生变化，如果赋值过小，元素就会丢失

2、indexOf，通过元素获得下标索引

字符串的 "1" 和数字的 1 是不同的

3、slice（）截取Array的一部分，返回一个新数组，类似与String中的substring

4、push，pop

- push：压入到尾部
- pop：弹出尾部的一个元素

5、unshift(),shift()头部

```
unshift: 压入到尾部
shift: 弹出尾部的一个元素
```

6、排序sort（）

7、元素翻转reverse（）

8、concat（）拼接数组

注意concat并没有修改数组，只是会返回一个新的数组

9、连接符join

打印拼接数组，使用特定的字符串连接

10、多维数组

数组：存储数据（如何存、如何取、方法都可以自己实现）

2.3、对象

若干键值对

```
let person={
            name : "kuangshen",
            age : 3,
            email : "957110695@qq.com",
            score : 5
        }
```

js中对象，{....}表示一个对象

js中的所有的键都是字符串，值是任意对象！

1、对象赋值

```
person.age="lwq"
'lwq'
person.age=3
3
```

2、使用一个不存在的对象属性，不会报错，undefined

3、动态的删减属性，通过delete删除对象的属性

```
delete person.age
true
person
{name: 'kuangshen', email: '957110695@qq.com', score: 5}
```

4、动态的添加

```
person.haha = 'haha'
'haha'
person
{name: 'kuangshen', email: '957110695@qq.com', score: 5, haha: 'haha'}
```

5、判断属性值是否在这个对象中！ xx in xx

```
`age` in person
false
// 继承
`toString` in person
true
```

6、判断一个属性是否是这个对象自身拥有的hasOwnProperty()

```
person.hasOwnProperty('toString')
false
person.hasOwnProperty('name')
true
```

2.4、流程控制

if判断

```
let age= 3
        if (age>3){
            alert("hh")
        }else {
            alert("no")
        }
```

循环

```
let age= 3
        while (age<100){
            age+=age
            console.log(age)
        }
do{
      age+=age
      console.log(age)
}
while (age<100)
```

for循环

```
let age= 3
        for (let i = 0; i < 100; i++) {
            console.log(i)
        }
```

foreach循环

```
var args=[1,2,4,67,74,4245,234,123]
        args.forEach(function (v) {
            console.log(v)
        })
```

for循环

```
//for(var index in object[])
var args=[1,2,4,67,74,4245,234,123]
        for (var num in args){
            console.log(args[num])
        }
```

3.5、Map和Set集合

Map：

```
let map = new Map([["tom",100],["jack",90],['gaga',25]]);

        let tom = map.get("tom");

        console.log(tom)

        map.set("admin",123456)
```

Set：无需不重复的集合

```
let set = new Set([3,1,1,1,1,1,1]);
        set.add(2)
        console.log(set.has(4))
```

3.6、iterator

```
//遍历数组
//通过for of遍历   for in 是下标
var arr = [1,2,3]
        for (var a of arr){
            console.log(a)
        }

//遍历map只能使用for of
let map = new Map([["tom",100],["jack",90],['gaga',25]]);
        for (var a of map){
            console.log(a)
        }
```

```
//遍历set
let set = new Set([3,1,1,1,1,1,1]);
        for (var a of set){
            console.log(a)
        }
```

3、函数

方法：在对象中存在（属性、方法）

函数：

本质是一样的只是位置不同

3.1、定义函数

定义方式一

```
public 返回值类型 方法名(){
    return 返回值;
}
//绝对值函数
function abs(x){
    if(x>=0){
         return x;
    }else{
         return -x;
    }
}
```

一旦执行到return代表函数结束，返回结果

如果没有执行return函数执行完也会返回结果，结果就是undefined

定义方式二

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

方式一和方式二等价

调用函数

```
abs(10)//10
abs(-10)//10
```

参数问题：js中可以传任意个参数，也可以不传递参数

arguments

arguments是js免费赠送的关键字；代表传递进来的所有的参数，是一个数组

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

问题：arguments包含所有的参数，我们有时候想使用多余的参数来进行附加操作，需要排除已有参数

rest

以前

```
function abs(a,b){
            console.log(a)
            console.log(b)
            if (arguments.length>2){
                for (var i =2;i<arguments.length;i++){
                    console.log(arguments[i])
                }
            }
        }
```

ES6引入的新特性，获取除了已经定义的参数以外的所有参数

```
function abs(a,b,...rest){
            console.log(a)
            console.log(b)
            if (arguments.length>2){
                for (var i =2;i<arguments.length;i++){
                    console.log(arguments[i])
                }
            }
        }
```

3.2、变量作用域

在javascript中，var定义变量实际是有作用域的

假设在函数体中声明，则在函数体外不可使用

```
function qj(){
            var x=1;
            x+=1;
        }
        x=x+2;//x is not defined
```

如果两个函数使用了相同的变量名，只要在函数内部，就不冲突

```
function qj(){
            var x=1;
            x+=1;
        }

        function qj2(){
            var x=1;
            x+=1;
        }
```

内部函数可以访问外部函数的成员，反之则不行

```
function qj(){
            var x = 1;
            function qj2() {
               var y= x + 1;
            }
            var z= y+1;
        }
```

假设，内部函数变量和外部函数的变量重名

```
function qj(){
            var x = 1;
            function qj2() {
               var x = 'A'
               console.log('inner'+x);
            }
            console.log('outer'+x);
            qj2();
        }
qj2()
```

假设在js中函数查找变量从自身函数开始，由内向外查找，假设外部存在同名的函数变量，则内部函数会屏蔽外部函数的变量

提升变量的作用域

```
function qj(){
    var x ='x'+y;
    console.log(x);
    var y = 'y';
}
```

结果：xundefined

说明：js 执行引擎，自动提升了y的作用域，但是不会提升变量y的赋值

```
function qj(){
    var y;
    var x ='x'+y;
    console.log(x);
    y = 'y';
}
```

这个是在js建立之初就存在的特性，养成规范，所有的变量定义都放在函数的头部，不要乱放，便于代码维护

全局变量

```
//全局变量
var x=1;
function f(){
    console.log(x)
}
console.log(x)
```

全局对象window

```
var x='xxx';
        alert(x)
        alert(window.x)
```

alert()函数也是一个window变量

任何变量（函数也可以视为变量），假设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找到，报错RefrenceError

规范

由于我们所有的全局变量都会绑定到我们的window上，如果不同的js文件，使用了 相同的全局变量，冲突-》如果能减少冲突

```
//唯一全局变量
var kuang={}
//定义全局变量
    kuang.name='kuangshen'
    kuang.add=function (a,b){
    return a+b;
}
```

把自己的代码全部放入自己定义的唯一全局变量

局部定义作用域let

let，解决局部作用域冲突问题

```
function a(){
            for (let i =0 ; i<100 ; i++){
                console.log(i)
            }
            console.log(i+1)
        }
```

建议使用let定义局部变量

常量const

ES6之前，只有用全部大写字母的叫做常量

```
var PI = 3.14
        console.log(PI);
        PI = 2.1
        console.log(PI)
```

ES6引入了关键字const

```
const PI = 3.14
        console.log(PI);
        PI = 2.1
        console.log(PI)
```

3.3、方法

定义方法

方法就是把函数放在对象的里面，对象只有两个东西：属性和方法

```
var kuangshen = {
            name : "kuangshen",
            birth : 2012,
            age : function (){
                var y = new Date().getFullYear();
                return y - this.birth;
            }
        };
//kuangshen.name 属性
kuangshen.age()//方法一定要带（）
```

this.代表什么？

```
function getAge(){
            var y = new Date().getFullYear();
            return y - this.birth;
        }

        var kuangshen = {
            name : "kuangshen",
            birth : 2012,
            age : getAge
        };
```

this是无法指向的，是默认指向调用它的那个对象；

apply

在js中可以控制this指向

```
function getAge(){
            var y = new Date().getFullYear();
            return y - this.birth;
        }

        var kuangshen = {
            name : "kuangshen",
            birth : 2012,
            age : getAge
        };
getAge.apply(kuangshen,[])//this，指向了kuangshen，参数为空
```

4、内部对象

标准对象

```
typeof 123
'number'
typeof '123'
'string'
typeof true
'boolean'
typeof NaN
'number'
typeof []
'object'
typeof {}
'object'
typeof Math.abs
'function'
typeof undefined
'undefined'
```

4.1、Date

基本使用：

```
let date = new Date();
        date.getDate()//日
        date.getMonth()//月 0-11代表月
        date.getFullYear()//年
        date.getDay()//星期几
        date.getTime()//时间戳 全世界统一 1970.1.1 0:00:00
        date.getMinutes()//分
        date.getHours()//时
```

```
let date = new Date();
        console.log(date)//Wed Sep 08 2021 11:24:12 GMT+0800 (中国标准时间)
```

```
//时间转换
date.toDateString()
'Wed Sep 08 2021'
date.toGMTString()
'Wed, 08 Sep 2021 03:35:18 GMT'
date.toJSON()
'2021-09-08T03:35:18.554Z'
date.toLocaleString()
'2021/9/8 上午11:35:18'
date.toLocaleDateString()
'2021/9/8'
date.toLocaleTimeString()
'上午11:35:18'
//注意调用的是一个方法
```

4.2、JSON

json是什么

早期的所有数据传输习惯都是使用xml文件传输

- 是一种轻量级 的数据交换格式
- 简洁和清晰的层次结构使得 JSON 成为理想的数据交换语言
- 易于人阅读和编写，同时也易于机器解析和生成，并有效地**提升网络传输效率**

在js一切皆为对象，任何js支持的类型都可以用JSON来表示

格式

- 对象{}
- 数组[]
- 所有的键值对 都是用 key:value

JSON 字符串和js对象的转化

```
var user={
            name : "1232",
            age : 3,
            sex : 'man'
        }
        let s = JSON.stringify(user);
        console.log(s)
        let parse = JSON.parse(s);
        console.log(parse)
```

JSON本身就是一个字符串

4.3、Ajax

- 原生的js写法，xhr异步请求
- jQuery封装好的方法$("#name").ajax("")
- axios请求

5、面向对象编程

5.1、什么是面向对象

javascript、java、c#。。。面向对象；javascript有些区别

- 类：模板
- 对象：具体的实例

在javascript中这个需要换一下思维

```
var user={
            name : "1232",
            age : 3,
            run : function (){
                console.log(this.name+"run..")
            }
        }
        var xiaoming={
            name : "xiaoming",
        }
//小明的原型指向了user
        xiaoming.__proto__=user;
```

class继承

class关键字是在ES6引入的

1、定义一个类，方法、属性

```
function Student(name){
            this.name = name
        }

        Student.prototype.hello=function (){
            alert("hello")
        }
        //=======ES6之后=======
        //定义一个学生的类
        class Student{
            constructor(name) {
                this.name=name;
            }
            hello(){
                alert("hello")
            }
        }

        var xiaoming = new Student("xiaoming")
```

2、继承

```
class Student{
            constructor(name) {
                this.name=name;
            }
            hello(){
                alert("hello")
            }
        }
```

```
        class xiaoxuesheng extends Student{
            constructor(name,age) {
                super(name);
                this.age=age;
            }
            myGrade(){
                alert("我是小学生")
            }
        }
        var xiaoming = new Student("xiaoming")
        var xiaogong = new xiaoxuesheng("xiaogong",3)
```

原型链

proto

6、操作BOM对象（重点）

浏览器介绍

js诞生就是为了能够让他在浏览器中运行

BOM：浏览器对象模型

- IE 6-11
- Chrome
- Safari （苹果）
- FireFox （Linux默认）
- Opera

三方

- QQ浏览器
- 360浏览器

window

window代表 浏览器窗口

```
window.alert(1)
undefined
window.innerHeight
968
window.innerWidth
940
window.outerHeight
1040
```

Navigator

Navigator，封装了浏览器的信息

```
navigator.appName
'Netscape'
navigator.appVersion
'5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/93.0.4577.63 Safari/537.36 Edg/93.0.961.38'
navigator.userAgent
'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/93.0.4577.63 Safari/537.36 Edg/93.0.961.38'
navigator.platform
'Win32'
```

大多数时候我们不会使用navigator对象，因为会被人为修改

不建议使用这些属性来判断和编写代码

screen

```
screen.width
1920
screen.height
1080
```

location

location代表当前页面的URL信息

```
host: "localhost:63342"
href: "http://localhost:63342/javaGame/com/lwq/juc/1.html?
_ijt=73uc2ku3td4fbla6h34e9oek9b"
protocol: "http:"
reload: ƒ reload()
//设置新的地址
location.assgin('https://www.baidu.com/')
```

document

document代表当前页面，HTML DOM文档树

```
document.title
'百度一下，你就知道'
document.title='nima'
'nima'
```

获取具体的文档树节点

```
<dl id="aa">
    <dt>java</dt>
    <dd>javase</dd>
    <dd>javaee</dd>
</dl>
<script>
    let dl = document.getElementById("aa");
</script>
```

获取cookie

document.cookie

```
<script src="aa.js"></script>
<!--恶意人员，获取你的cookie上传到他的服务器-->
```

服务器端可以设置cookie：httpOnly来保证安全性

history

```
history.back();后退
history.forward();前进
```

7、获得DOM对象（重点）

DOM：文档对象模型

核心

浏览器网页就是一个Dom树形结构

更新：更新Dom节点

遍历Dom节点：得到Dom节点

删除：删除一个Dom节点

添加：添加一个新的节点

要操作一个Dom节点，就必须要先获得这个Dom节点

```
获得dom节点

<div id="father">
    <h1>1</h1>
    <p id="p1">p1</p>
    <p id="p2">p2</p>
</div>

<script>
    let h1 = document.getElementsByTagName('h1');
    let p1 = document.getElementById('p1');
    let p2 = document.getElementsByClassName('p2');
    let father = document.getElementById("father");

    let children = father.children;//获取父节点下的所有子节点
    // father.firstChild
    // father.lastChild
</script>
```

之后尽量使用jquery

更新节点

```
id1.innerText='123'
'123'
id1.innerText='456'
'456'

操作文本，操作js
id1.innerHTML='<strong>123</strong>' //可以解析HTML文本标签
id1.style.color='red'//属性使用字符串包裹
id1.style.fontSize='120px'//转驼峰命名
```

删除节点

```
通过父节点删除子节点
father.removeChild(p1)

//删除是一个动态的过程
father.removeChild(father.children[0]);
father.removeChild(father.children[1]);
father.removeChild(father.children[2]);
```

注意：删除多个节点的时候，children是在时刻变化的，删除节点的时候一定要注意！

插入节点

我们获得了某个dom节点

追加

```
<p id="js">
    JavaScript
</p>
<div id="list">
    <p id="se">javaSE</p>
    <p id="ee">javaEE</p>
    <p id="me">javaME</p>
</div>

<script>
    "use strict";
    let js = document.getElementById("js");
    let me = document.getElementById("me");
    me.append(js);
</script>
```

创建一个节点

```
<script>
    "use strict";
    let js = document.getElementById("js");
    let me = document.getElementById("me");
    me.append(js);
    let p = document.createElement('p');//创建一个空的节点
    p.id='newp';
    p.innerText="hello";
</script>

let script = document.createElement('script');

    script.setAttribute('type','text/css');//通过这个属性可以创建任意属性
```

insertbefore（oldNode，newNode）

8、操作表单（验证）

表单是什么 from DOM树

- 文本框 text
- 下拉框 select
- 单选框 radio
- 多选框 checkbox
- 隐藏域 hidden
- 密码框 password

表单的目的：提交信息

```
<form action="post">
    用户名：<input type="text" id="username">
    性别：<input type="radio" name="sex" value="man" id="boy">男
    <input type="radio" name="sex" value="woman" id="girl">女
    爱好：<label>
    <input type="checkbox">
</label>
</form>
<script>
    "use strict";
    let username = document.getElementById("username");
    username.value="qqq"//修改输入框的值

    let boy_radio = document.getElementById("boy");
    let girl_radio = document.getElementById("girl");

    boy_radio.checked//查看返回结果，是否为true，如果为true则被选中
    boy_radio.checked="true"//赋值
</script>
```

提交表单

```
<script src="https://cdn.bootcss.com/blueimp-md5/2.10.0/js/md5.min.js"></script>
<form action="#" method="post">
    用户名：<input type="text" id="username" name="username">
    密码：<input type="password" id="input-password" >
```

```
    <input type="hidden" name="password" id="md5-password">
    <!--绑定事件-->
    <button type="submit" onclick="aaa()">提交</button>
</label>
</form>
<script>
    "use strict";
    function  aaa(){
        let username = document.getElementById("username");
        let password = document.getElementById("input-password");
        let md5pwd = document.getElementById("md5-password");
        console.log(username.value);
        md5pwd.value=md5(password.value);

        console.log(password.value);
    }

</script>
```

9、jQuery

javascript

jquery：库，里面存在着大量的javascript函数

公式：$(selector).action()

```
<script src="https://lib.baomitu.com/jquery/3.6.0/jquery.js"></script>
$('#test-jquery').click(function (){
        alert("hello")
    })
```

```
$('p').click();
   $('#id').click();
   $('.class').click();
```

事件

鼠标事件，键盘事件

```
$(function (){
    $('#divmove').mousemove(function (e){
        $('#mousemove').text('x:'+e.pageX+'y:'+e.pageY)
    })
})
```

操作Dom元素

```
$('#test-ul li[name = python]').text();//取得值
$('#test-ul li[name = python]').text("12300");//设置值
```

css的操作

```
$('#test-ul li[name = python]').css({"color":"red","fontSize":"200px"});
```

元素的显示和隐藏：本质display：none；

未来ajax

```
$.ajax({url:"test.html",context:document.body,success:function (){
        $(this).addClass("done")
        }})
```