JavaScript基础

今日内容

> 1.掌握JavaScript的基础语法
>
> 2.会使用JavaScript常见数据类型
>
> 3.掌握JavaScript函数的语法
>
> 4.掌握JS事件绑定方式
>
> 5.能够使用常用JS内置对象

第1章 JavaScript概述

JS简介

1.1 JS作用

> JS用于完成页面与用户的交互功能.

1.2 JS发展历史

> 布兰奇 艾奇 在1995年的时候使10天时间发明了JavaScript语言.
>
> 欧洲计算机制造联盟(ECMA) 在1997制定脚本语言规范 ECMA Script1 (es1)
>
> 2009年发布了ECMA Script5 (es5) ,在2015年发布了ECMA Script2015 (es6)
>
> 所有的浏览器的都支持es6

1.3 JS的特点

JS设计模仿了java语言,列举不同

> 1. JS不需要编译,由浏览器直接解释执行.
> 2. js是弱类型语言,js变量声明不需要指明类型,不同类型的数据可以赋值给同一变量.

1.4 JS组成

> ECMA Script js的基础语法
>
> BOM(Brower Object Model) 浏览器对象模型
>
> DOM(Document Object Model) 文档对象模型

第2章 JavaScript基础语法

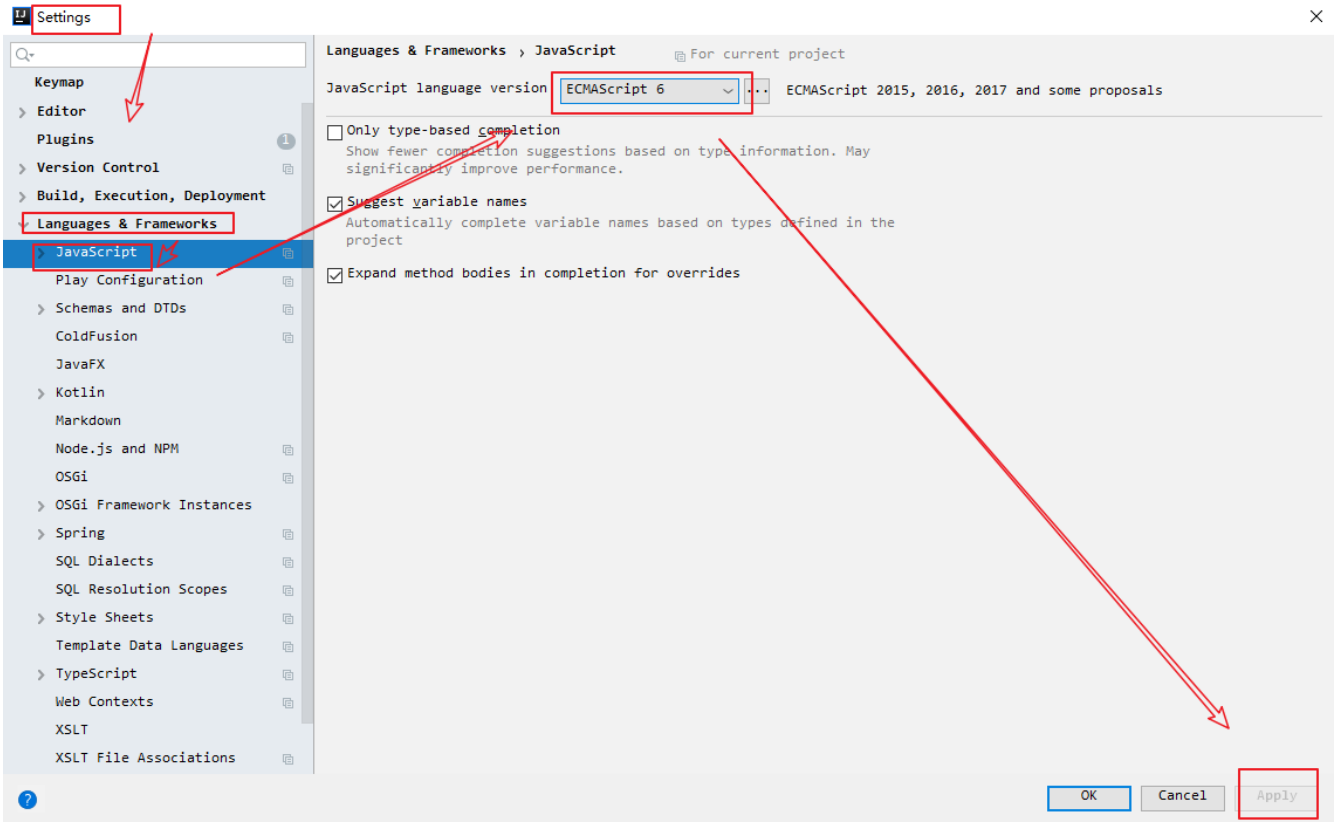
2.1-JS两种引入方式

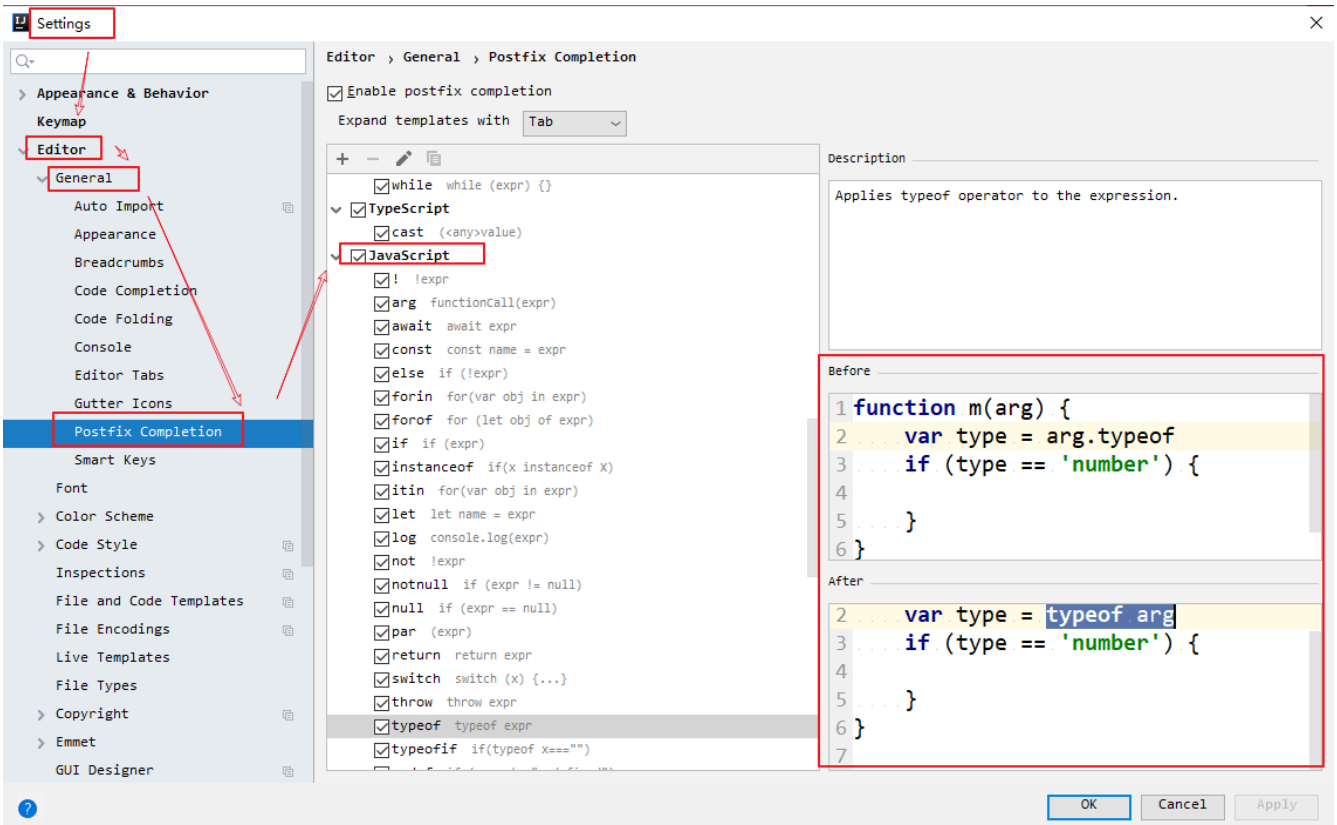

```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>JS两种引入方式</title>

</head>
<body>
<!--
JS两种引入方式
    JS和CSS一样都需要引入到html页面中,浏览器才会解释执行
    JS有两种引入方式
        1. 内嵌(内部)脚本:在script标签中写js代码,
            script标签推荐放置在body标签的底部,理论上和style标签一样可以放置位置任意.
        2. 外部脚步: 使用script标签的src属性引入外部js文件,使用注意事项:
            (没有内容还不能自闭合):
            script标签使用了src属性以后内部的代码不再被浏览器解释执行(了解).
            script引入外部脚本时不能使用自闭合格式(警告!!!).

-->

<!--告诉浏览器把解析器切换为js解析器  type="text/javascript"可以省略-->
<script type="text/javascript">
    document.write("<h1>内部脚本</h1>");//向body中追加html内容
</script>

<script src="../js/外部脚本.js"></script>




</body>
</html>
```

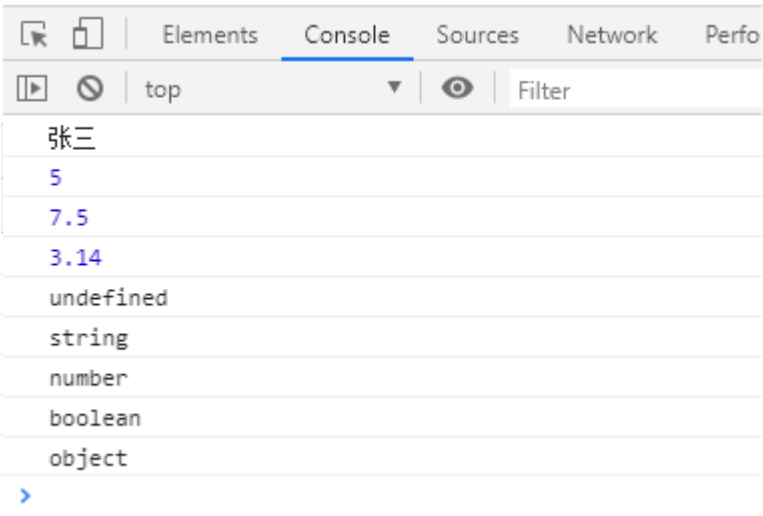
2.2-JS三种输出方式


```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>JS三种输出方式</title>
</head>
<body>
<!--
JS三种输出方式
    1. 输出到浏览器控制台
    2. 输出html内容到页面
    3. 浏览器弹框输出字符
 -->

<script >
    //1. 输出到浏览器控制台
    console.log("1. 输出到浏览器控制台");//开发者专用
    //2. 输出html内容到页面
    document.write("2. 输出html内容到页面");//向body中追加html内容
    //3. 浏览器弹框输出字符
    alert("3. 浏览器弹框输出字符");//阻塞执行
</script>
</body>
</html>
```

2.3-JS变量声明


Settings
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Languages & Frameworks
JavaScript
Play Configuration
Schemas and DTDs
ColdFusion
JavaFX
Kotlin
Markdown
Node.js and NPM
OSGi
OSGi Framework Instances
Spring
SQL Dialects
SQL Resolution Scopes
Style Sheets
Template Data Languages
TypeScript
Web Contexts
XSLT
XSLT File Associations
Languages & Frameworks › JavaScript
For current project
JavaScript language version
ECMAScript 6
ECMAScript 2015, 2016, 2017 and some proposals
Only type-based completion
Show fewer completion suggestions based on type information. May significantly improve performance.
Suggest variable names
Automatically complete variable names based on types defined in the project
Expand method bodies in completion for overrides
OK
Cancel
Apply

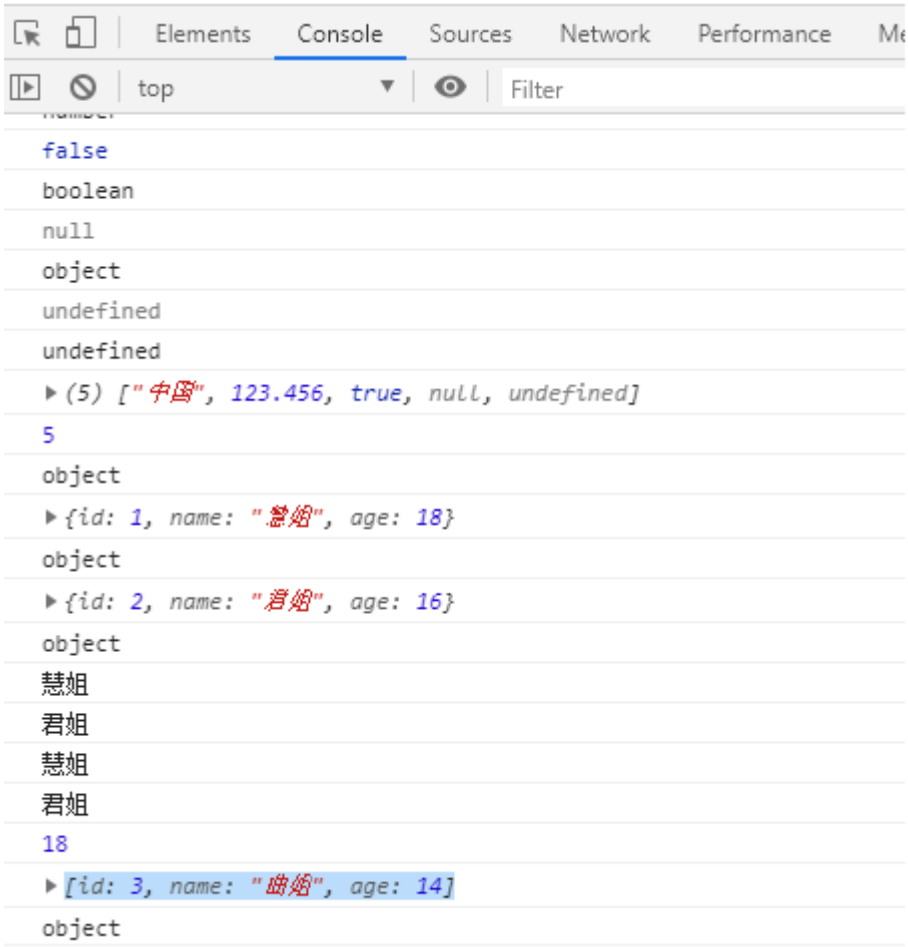

Settings
Appearance & Behavior
Keymap
Editor
General
Auto Import
Appearance
Breadcrumbs
Code Completion
Code Folding
Console
Editor Tabs
Gutter Icons
Postfix Completion
Smart Keys
Font
Color Scheme
Code Style
Inspections
File and Code Templates
File Encodings
Live Templates
File Types
Copyright
Emmet
GUI Designer
Editor › General › Postfix Completion
Enable postfix completion
Expand templates with
Tab
while while (expr) {}
TypeScript
cast (<any>value)
JavaScript
! !expr
arg functionCall(expr)
await await expr
const const name = expr
else if (!expr)
forin for(var obj in expr)
forof for (let obj of expr)
if if (expr)
instanceof if(x instanceof X)
itin for(var obj in expr)
let let name = expr
log console.log(expr)
not !expr
notnull if (expr != null)
null if (expr == null)
par (expr)
return return expr
switch switch (x) {...}
throw throw expr
typeof typeof expr
typeofif if(typeof x==="")
Description
Applies typeof operator to the expression.
Before
function m(arg) {
var type = arg.typeof
if (type == 'number') {
}
}
After
var type = typeof arg
if (type == 'number') {
}
}
OK
Cancel
Apply




```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>JS变量</title>
</head>
<body>
<!--
JS变量 是用来存放数据
    es6之后变量声明使用let 常量声明使用const 他们用于替代es6的var声明方式
    (了解) es5以前变量声明使用var

    JS是弱类型语言(理解):
        声明变量时不知道变量的类型(undefined),只有在赋值之后js变量才确定类型.
        typeof(a) 或 typeof a 输出变量的类型
        undefined 变量未赋值,未知类型



-->
<script >
    //字符串 Java声明 String str ="张三";
    let str ="张三";
    console.log(str);
    //整数  Java声明 int k = 5;
    let k = 5;
    console.log(k);
    //小数 Java声明 float f = 7.5;
    let f = 7.5;
    console.log(f);
    //常量 Java声明 final Integer PI = 3.14;
    const PI = 3.14;
    console.log(PI);
    //演示弱类型语言
    let a;//声明变量不需要指明类型
    console.log(typeof a);//undefined 未赋值类型,未知类型
```

```
    a = "你好";
    console.log(typeof a);//string
    a = 123;
    console.log(typeof a);//number
    a = true;
    console.log(typeof a);//boolean
    a = new Object();
    console.log(typeof a);//object


</script>
</body>
</html>
```

2.4-JS数据类型简介




```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>JS数据类型</title>
```

```
</head>
<body>
<!--
JS数据类型
    常用数据类型
        1. string 字符串类型
        2. number 数字 包括整数和小数类型
        3. boolean 布尔类型 值只有 true和false 两个
        4. object 对象类型 空对象使用 null表示
                有两种格式(了解)
                    Object格式 例如 new Object();
                    JSON格式 例如 {name:"张三",age:18}
        5. undefined 变量未赋值
-->
<script >
    //1. string 字符串
    let str = "你好";
    console.log(str);
    console.log(typeof str);//string
    //2. number 数字
    let n = 123.456;
    console.log(n);
    console.log(typeof n);//number
    //3. boolean 布尔类型
    let boo = false;
    console.log(boo);
    console.log(typeof boo);//boolean
    //4. object 对象类型 空对象使用 null表示
    let obj = null;//或 new Object();
    console.log(obj);
    console.log(typeof obj);//object
    //5. undefined 变量未赋值
    let u = undefined;
    console.log(u);//值是undefined
    console.log(typeof u);//类型是undefined


    //Object类型
    let stu = new Object();//创建一个js对象,js对象的属性想要直接加上
    stu.id = 1;
    stu.name = "刘一";
    stu.age = 18;
    console.log(stu);//{id: 1, name: "刘一", age: 18}
    console.log(typeof stu);//object


    //  JS对象取属性值有两种方式:
    //1. obj.key
    console.log(stu.name);//刘一
    console.log(per.name);//陈二

    //2. obj["key"]
    console.log(stu["name"]); //刘一 == stu.name
```

```
    console.log(per["name"]);//陈二 == per.name
    let b = "age";
    console.log(stu[b]);//可以取不定属性的值

</script>
</body>
</html>
```

2.5-JS运算符

JavaScript 算术运算符

算术运算符用于执行变量与/或值之间的算术运算。

给定 **y=5**，下面的表格解释了这些算术运算符：

运算符	描述	例子	结果
+	加	x=y+2	x=7
-	减	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	求余数（保留整数）	x=y%2	x=1
++	累加	x=++y	x=6
--	递减	x=--y	x=4

JavaScript 赋值运算符

赋值运算符用于给 JavaScript 变量赋值。

给定 **x=10** 和 **y=5**，下面的表格解释了赋值运算符：

运算符	例子	等价于	结果
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

比较运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等。

给定 x=5，下面的表格解释了比较运算符：

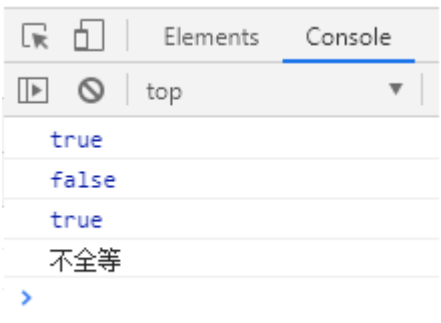
运算符	描述	例子
==	等于	x==8 为 false
===	全等（值和类型）	x===5 为 true；x==="5" 为 false
!=	不等于	x!=8 为 true
>	大于	x>8 为 false
<	小于	x<8 为 true
>=	大于或等于	x>=8 为 false
<=	小于或等于	x<=8 为 true

逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符：

运算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
||	or	(x==5 || y==5) 为 false
!	not	!(x==y) 为 true

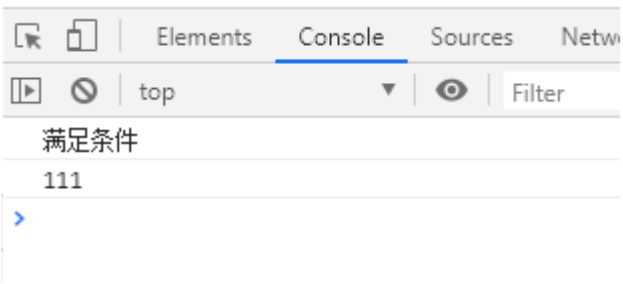



```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>JS运算符</title>
</head>
<body>
<!--
JS运算符
    因为JS是参照Java而设计的语言，所以js运算符和Java运算符基本相同
```

```
    只有一个特殊的比较运算符
    === 判断js变量的值和类型都相等才为true
    !== 判断js变量的值和类型有一个不等就为true
-->

<script >
    let a = 3;
    let b = "3";
    console.log(a == b);//true
    //全等 运算符 ===
    console.log(a === b);//false
    //不全等 运算符 !==
    console.log(a !== b);//true
    // 三元(三目)运算符  布尔表达式?真:假
    let str = a===b?"全等":"不全等";
    console.log(str);//不全等

</script>
</body>
</html>
```

JS流程控制语句

2.6-条件语句




```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>条件语句</title>
</head>
<body>
<!--
条件语句
    JS的条件语句和Java语法基本一样,但是对数据类型的真假判断有些区别
    JS中对各种数据类型作为布尔值的特点:(重点掌握)
        1. string 空字符串""为false,其余都为true
        2. number 数字 只有0为false,其余数字都为true
        3. boolean 布尔类型 值只有 true和false 两个
        4. object 对象类型 空对象null表示false,其它对象都是true
```

```
        5. undefined 变量未赋值 为false
    常用语法格式
        if ... else if ... else
        switch case break default

-->
<script >
    //if ... else if ... else
    // if(true){
    // if(""){//string 只有空字符为假
    // if(0){number 只有0为假
    // if(false){//boolean false为假 true为真
    // if(null){//object null为假
    // if(undefined){//undefined永为假
    if("undefined"){//undefined永为假
        console.log("满足条件");
    }else{
        console.log("不满足条件");
    }



    //switch case break default

    let k =1;
    switch (k) {
        case 1:
            console.log("111");break;
        case 2:
            console.log("222");break;
        default:
            console.log("其它情况");
    }


</script>
</body>
</html>
```

2.7-循环语句


```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>循环语句</title>
</head>
<body>
<!--
循环语句
    while 和Java一样
    do while 和Java一样
    (重点掌握:)
    fori 和Java一样
    forin 1.遍历出数组中的索引 2.遍历出对象中的属性名key
    forof 1.遍历出数组中的元素

重点记忆点:
    forin 与 forof 区别:
        1.forin可以遍历对象,forof不能遍历对象
        2.forin遍历出数组中的索引,forof遍历出数组中的元素

-->
<script >
    //while 和Java一样
```

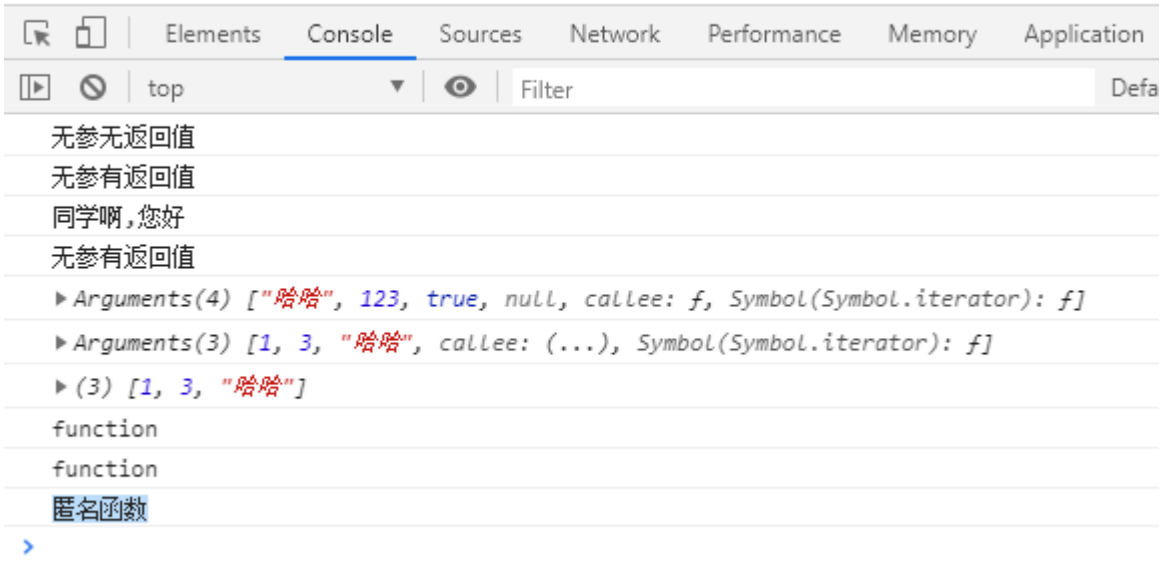
```
    let k=1;
    while (k<3){
        console.log(k++);
    }

    //do while 和Java一样
    k =1;
    do{
        console.log(k++);
    }while (k<3)



    //fori 和Java一样
    for(let i=0;i<3;i++){
        console.log(i);
    }

    // forin 可以遍历数组和对象
    let arr = ["刘一","陈二","张三"];//JS数组使用中括号[]定义
    let stu = {id:5,name:"李四",age:18};//JS对象使用大括号定义
    //forin  1.遍历出数组中的索引
    for(let index in arr){
        console.log(index);//数组的索引 0,1,2
        console.log(arr[index]);//数组中的元素
    }

    //forin  2.遍历出对象中的属性名key
    for(let k in stu){
        console.log(k);//字符串属性 id,name,age
        console.log(stu[k]);//对象中的属性值
    }

    //forof 1.遍历出数组中的元素
    for(let e of arr){
        console.log(e);//数组中的元素
    }


</script>
</body>
</html>
```

第3章 JS函数

3.1-JS函数

函数

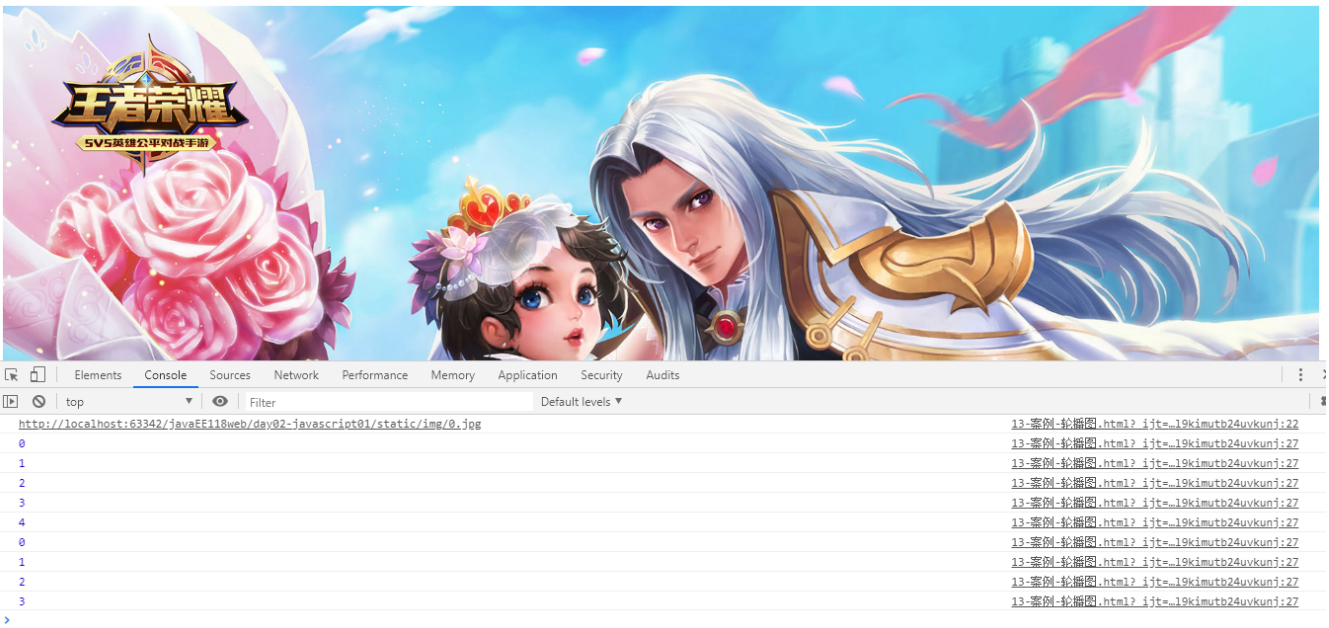
> js函数是执行特定功能的代码块.也可以称为js方法




```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>JS函数</title>
</head>
<body>
<!--
JS函数 使用关键字 function 来声明
    1. 普通函数 格式 function 函数名(参数){return}
    2. 匿名函数 格式 function(参数){return}
注意
    a.普通函数和匿名函数中都有一个参数数组对象 arguments
    b. JS函数的调用只以函数名区分,与参数无关
    c. JS函数也是一种类型,一个具体函数可以看做是一个函数类型变量的一个值

-->
<script >

//------------- 普通函数 ------------------
//  function myf( name,age){
//     return "我是函数";
// }
    //无参无返回值
function f1(){
    console.log("无参无返回值");
}
f1();

    //无参有返回值
```

```
function f2(){
    return "无参有返回值";
}
let s2 = f2();//方法调用,并接收返回值
console.log(s2);
//有参有返回值
function f3(name){
    return name+",您好";
}
let s3 = f3("同学啊");////方法调用,并接收返回值
console.log(s3);

//函数调用 js函数没有重载的概念. js方法的调用只以方法名区分
s2 = f2(1,2,3);//方法调用,并接收返回值
console.log(s2);

    //js中每一个方法都有一个arguments参数数组(了解)
function f22(){
    console.log(arguments);
    return "无参有返回值";
}
f22("哈哈",123,true,null);//调用f22方法执行,并传入四个参数
    //接受不定参数 ES6(了解)
function f4(... strs){
    console.log(arguments);//参数数组
    console.log(strs);//接收不定参数,同上
}
f4(1,3,"哈哈");//调用f4方法,并传入参数
//------------- 匿名函数 ------------------
    //匿名函数
    let f5 = function (){
        return "匿名函数";
    };
console.log(typeof f5);//function
console.log(typeof f1);//function

console.log(f5());//调用f5


</script>
</body>
</html>
```

3.2-案例:轮播图




```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>案例-轮播图</title>
</head>
<body>
<!--
轮播图

预习:
    setInterval(函数,间隔时间) 每隔固定间隔时间(毫秒)执行一次函数
    document.querySelector(css选择器) 根据css选择器获取匹配到的一个标签

-->
<img src="../img/0.jpg" width="100%">

<script >

    let img = document.querySelector("img");//获取页面中的第一个img标签对象
    console.log(img);//<img src="../img/0.jpg" width="100%">
    console.log(img.src);//取出src属性的值 ../img/0.jpg

    let k =0;
    function myf(){
        img.src = "../img/"+k+".jpg";//给img标签对象的src属性赋值
        console.log(k++);
        if(k==5)k=0;
    }
    setInterval(myf,1000);//创建一个定时器对象,每隔1秒调用一次myf,直到永远


</script>
</body>
```

```
</html>
```

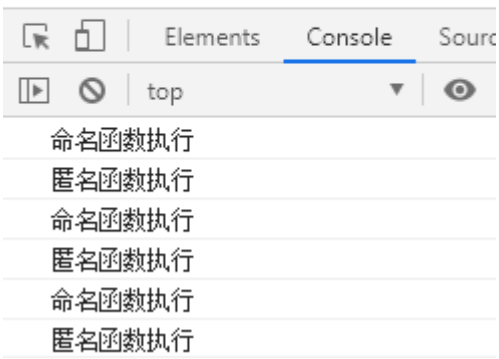
第4章 JS事件

> JS可以监听用户的行为,并调用函数来完成用户交互功能.

4.1-事件的绑定方式(重点)

---

1. 命名函数
2. 匿名函数



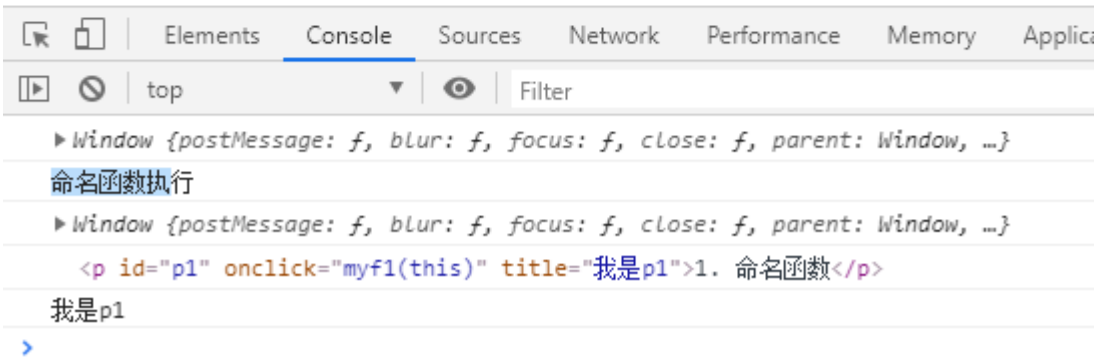

```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>事件的绑定方式</title>
</head>
<body>
<!--
事件的绑定方式
```

```
    1. 命名函数 格式 onEvent=函数
    2. 匿名函数 格式 onEvent=函数
-->

<p id="p1" onclick="myf1()" title="我是p1">1. 命名函数</p>
<p id="p2" title="我是p2">2. 匿名函数</p>

<script >
    //1. 命名函数 格式 onEvent="函数()"
    function myf1(){//命名函数绑定把方法调用给事件属性
        console.log("命名函数执行");
    }

    //2. 匿名函数 格式 onEvent=函数
    let p2 = document.querySelector("#p2");//获取p2
    p2.onclick = function(){//当p2被点击的时候调用方法执行
        console.log("匿名函数执行");
    };


</script>
</body>
</html>
```

4.2-两种绑定方式区别

1. 命名函数

2. 匿名函数



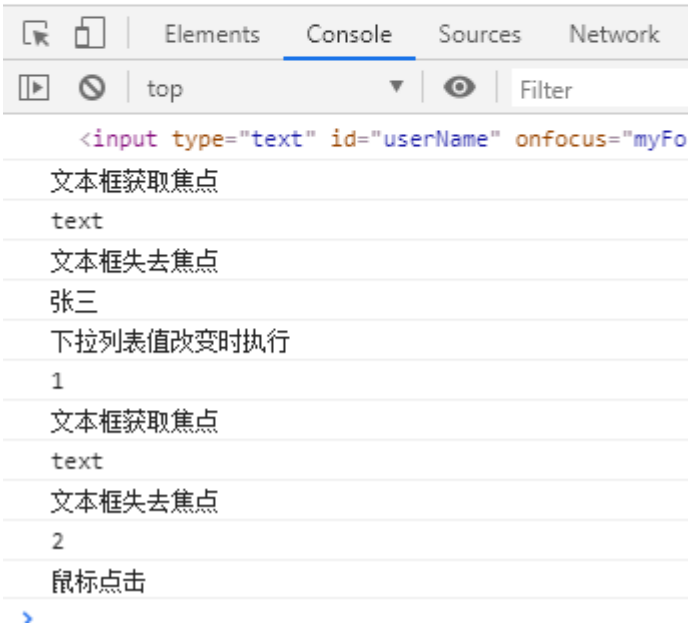

```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>两种绑定方式区别</title>
</head>
<body>
<!--
事件的绑定方式
    1. 命名函数 没有this绑定,需要手动传递this
    2. 匿名函数 默认有this绑定,可以直接使用
-->

<p id="p1" onclick="myf1(this)" title="我是p1">1. 命名函数</p>
<p id="p2" title="我是p2">2. 匿名函数</p>

<script >
    console.log(this);//window 浏览器对象

    //1. 命名函数 格式 onEvent="函数()"
```

```
    function myf1(obj){//命名函数绑定把方法调用给事件属性
        console.log("命名函数执行");
        console.log(this);//window 浏览器对象
        console.log(obj);//<p id="p1" onclick="myf1(this)" title="我是p1">1. 命名函数</p>
        console.log(obj.title);//我是p1
    }

    //2. 匿名函数 格式 onEvent=函数
    let p2 = document.querySelector("#p2");//获取p2
    p2.onclick = function(){//当p2被点击的时候调用方法执行
        console.log("匿名函数执行");
        console.log(this);//p2 == <p id="p2" title="我是p2">2. 匿名函数</p>
        console.log(this.title);//我是p2
    };


</script>
</body>
</html>
```

事件种类

onabort	图像的加载被中断。
onblur	元素失去焦点。
onchange	域的内容被改变。
ondick	当用户点击某个对象时调用的事件句柄。
ondblclick	当用户双击某个对象时调用的事件句柄。
onerror	在加载文档或图像时发生错误。
onfocus	元素获得焦点。
onkeydown	某个键盘按键被按下。
onkeypress	某个键盘按键被按下并松开。
onkeyup	某个键盘按键被松开。
onload	一张页面或一幅图像完成加载。
onmousedown	鼠标按钮被按下。
onmousemove	鼠标被移动。
onmouseout	鼠标从某元素移开。
onmouseover	鼠标移到某元素之上。
onmouseup	鼠标按键被松开。
onreset	重置按钮被点击。
onresize	窗口或框架被重新调整大小。
onselect	文本被选中。
onsubmit	确认按钮被点击。
onunload	用户退出页面。

4.3-常用事件(难点)

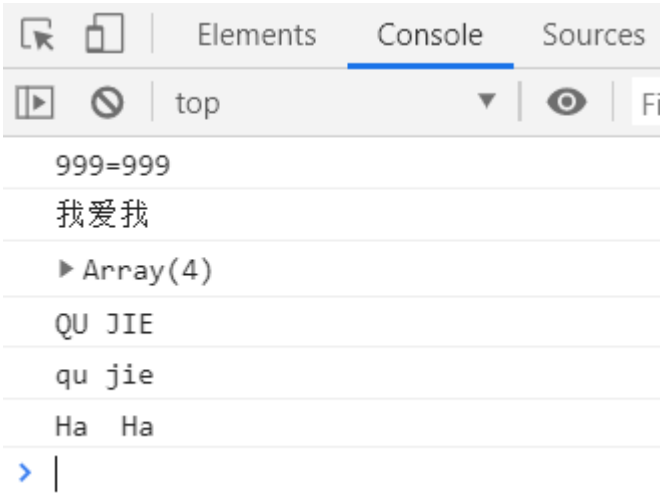



```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>常用事件</title>
</head>
<body>

<!--
常用事件
    1. onload 页面加载完成
    2. onfocus 获取焦点
    3. onblur 失去焦点
    4. onchange 表单控件的值改变时
    5. onclick 鼠标单击

-->

姓名 <input type="text" id="userName" onfocus="myFocus(this)"><br/>
学历
<select name="edu" id="edu">
    <option value="0">请选择</option>
```

```
    <option value="1">本科</option>
    <option value="2">大专</option>
</select>
<br/>
<button id="btn" onclick="myclick()">按钮</button>
<script >
    //1. onload 页面加载完成
    window.onload = function(){
        let userName = document.querySelector("#userName");//获取普通文本框对象
        console.log(userName);//<input type="text" id="userName">
    };

    //2. onfocus 获取焦点
    function myFocus(input){
        console.log("文本框获取焦点");
        console.log(input.type);//text

    }


    //3. onblur 失去焦点
    let myinput = document.querySelector("#userName");//获取普通文本框对象
    myinput.onblur = function (){//文本框失去焦点时执行
        console.log("文本框失去焦点");
        console.log(this.value);//文本框中输入的值
    };


    //4. onchange 表单控件的值改变时
    let select = document.querySelector("#edu");//获取下拉列表标签对象
    select.onchange=function(){//下拉列表值改变时执行
        console.log("下拉列表值改变时执行");
        console.log(this.value);//下拉列表的值
    }


    //5. onclick 鼠标单击
    function myclick(){
        console.log("鼠标点击");
        select.value = myinput.value;//把文本框的值赋值给下拉列表的值
    }


</script>
</body>
</html>
```

第5章 JS常用内置对象

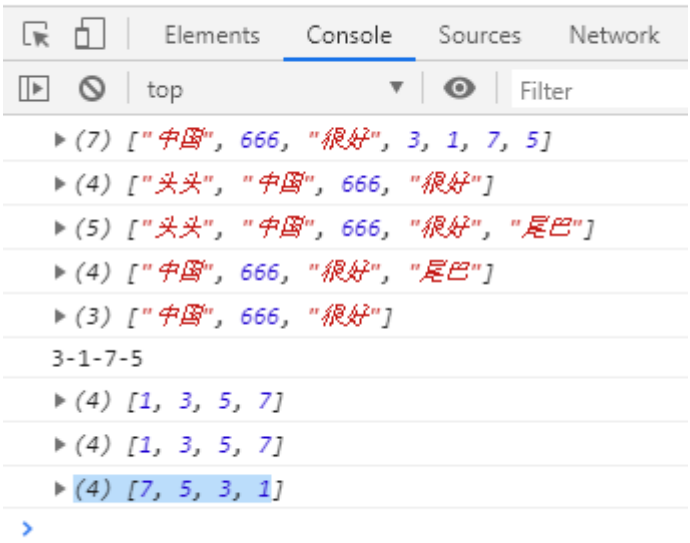
4.1-字符串




```
<!DOCTYPE html>
<html lang="zh">
<head>
  <meta charset="UTF-8">
  <title>字符串</title>
</head>
<body>
<!--
1. 构造字符串对象可以使用
    双引号,单引号,反引号

2. 字符串方法
    substring(startIndex,endIndex) 提取字符串中两个指定的索引号之间的字符,截取字符和slice相似
    split(delimiter) 把字符串分割为子字符串数组;分割为数组,反array.join(delimiter)
    toLowerCase() 把字符串转换为小写
    toUpperCase() 把字符串转换为大写
    trim() 移除字符串首尾空白
-->

<script type="text/javascript">
// ------------ 1. 构造字符串对象可以使用  双引号,单引号,反引号,new关键字
  //双引号字符串
let s1 = "双引号字符串";

  //单引号
let s2 = '单引号字符串';

  //反引号字符又叫做`字符串模板` ,模板中可以使用${}来进行插值.
let s3 = `反引号字符串`;
let n = 999;
let s4 = "999="+n;
let s5 = `999=${n}`;
console.log(s5);//"999=999"
```

```
// ------------ 2. 字符串方法

  //截取字符串
let s11 = "我是中国人,我爱我的祖国";
console.log(s11.substring(6, 9));//截取字符串,从索引6截取到9(不包括9)

//字符串分割为数组
let s12 = "1,2,3,4";
console.log(s12.split(","));//字符串分割为数组


//转换大小写
let s14 = "Qu Jie";
console.log(s14.toUpperCase());//大写
console.log(s14.toLowerCase());//小写

//去除首尾空格
  let s15 = "    Ha  Ha    ";
console.log(s15.trim());//去掉首尾空格

</script>
</body>
</html>
```

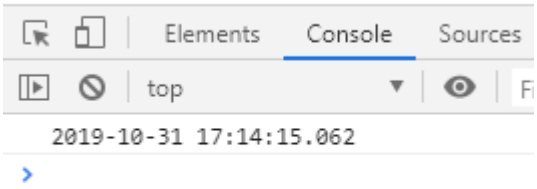
4.2-数组




```
<!DOCTYPE html>
<html lang="zh">
<head>
```

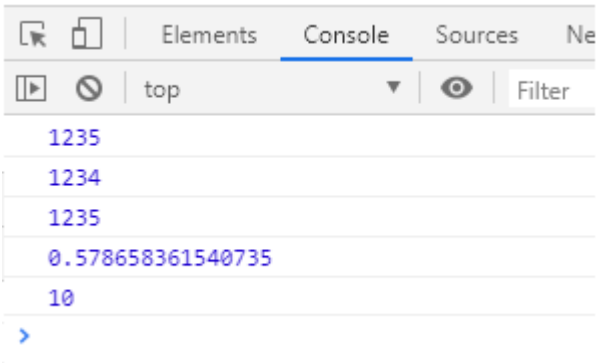
```
    <meta charset="UTF-8">
    <title>数组</title>
</head>
<body>
<!--

JS数组的特点:
    1.JS是弱类型,数组元素类型任意
    2.JS的数组类似于Java的集合,长度可变,所以有时又把js数组叫做数组或集合

数组
    要求能够查询w3c手册完成如下功能
        1. 创建数组 []
        2. 数组合并 concat
        3. 添加元素
            数组头添加 unshift
            数组尾添加 push
        4. 删除元素
            数组头删除 shift
            数组尾删除 pop
        5. 数组元素拼接为字符串 join(分隔符)
        6. 排序数组元素 sort

-->
<script >
    //1. 创建数组
    let arr1 = ["中国",666,"很好"];
    let arr2 = [3,1,7,5];
    //2. 数组合并
    console.log(arr1.concat(arr2));//数组合并
    //3. 添加元素
    arr1.unshift("头头");//头添加
    console.log(arr1);
    arr1.push("尾巴");//尾添加
    console.log(arr1);
    //4. 删除元素
    arr1.shift();//头删除
    console.log(arr1);
    arr1.pop();//尾删除
    console.log(arr1);
    //5. 数组元素拼接为字符串
    console.log(arr2.join("-"));//拼接  "3-1-7-5"
    //6. 排序数组元素
    console.log(arr2.sort());//排序 默认是升序
    console.log(arr2.sort(function(a,b){return a-b;}));//升序
    console.log(arr2.sort(function(a,b){return b-a;}));//降序


</script>
</body>
</html>
```

4.3-日期




```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>日期</title>
</head>
<body>
<!--
日期
    查询手册完成:
    1. 创建日期对象
        new Date() // 当前日期和时间
    2. 获取当前时间 转为字符串 格式 yyyy-MM-dd HH:mm:ss.SSS

-->

<script >
function getDateStr(){
    //1. 创建日期对象
    let d = new Date();//获取当前时间对象
    //2. 获取当前时间 转为字符串 格式 yyyy-MM-dd HH:mm:ss.SSS
    let fullYear = d.getFullYear();//获取年
    let month = new String(d.getMonth()+1).padStart(2,"0");//获取月 09
    let date = new String(d.getDate()).padStart(2,"0");//日
    let hours = new String(d.getHours()).padStart(2,"0");//时
    let minutes = new String(d.getMinutes()).padStart(2,"0");//分
    let seconds = new String(d.getSeconds()).padStart(2,"0");//秒
    let milliseconds = new String(d.getMilliseconds()).padStart(3,"0");//毫秒

    let dateStr = `${fullYear}-${month}-${date}
${hours}:${minutes}:${seconds}.${milliseconds}`;
    console.log(dateStr);//yyyy-MM-dd HH:mm:ss.SSS
    return dateStr;
}

getDateStr();//调用方法


</script>
</body>
</html>
```

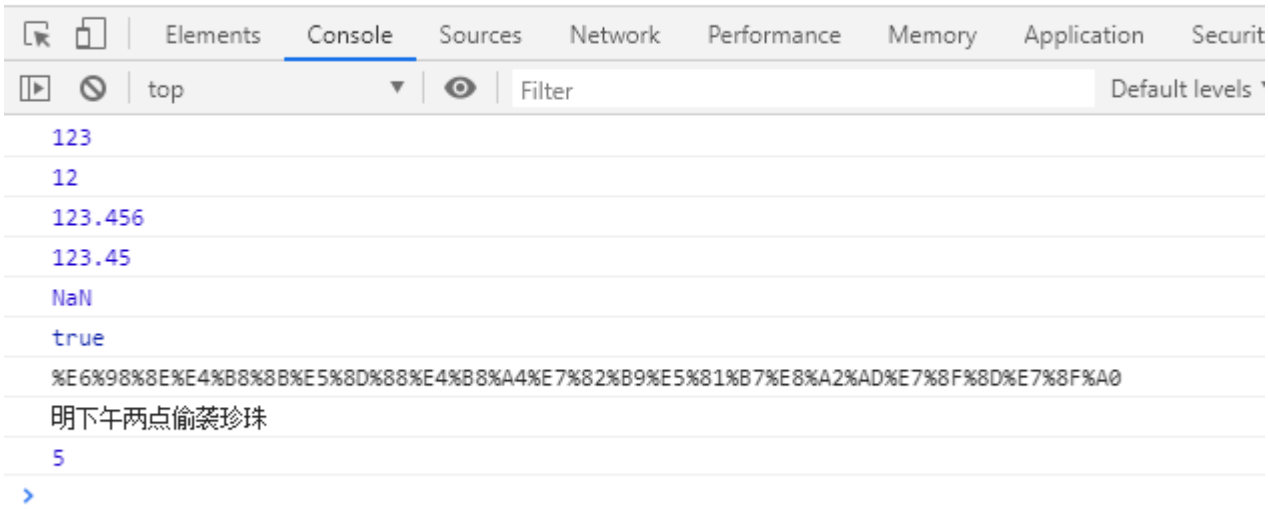
4.4-数学运算




```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>数学运算</title>
</head>
<body>
<!--
数学运算
    查询手册完成
    1. 四舍五入 round(x) 把数四舍五入为最接近的整数
    2. 向下取整 floor(x) 对数进行下舍入。 地板
    3. 向上取整 ceil(x) 对数进行上舍入。
    4. 产生随机数 random() 返回 0 ~ 1 之间的随机数。
    5. 产生 [1,10]的随机整数

小扩展:
    parseInt(x) 对x转为整数,有小数部分直接舍去. 功能类似 floor(x)
    random() 返回 [0,1) 之间的随机数。 [0,1) 左闭右开区间,包含0不包含1
    在任何语言的随机数中都是左闭右开.

-->
<script >
    let n = 1234.567
    //1. 四舍五入取整
    console.log(Math.round(n));//1235
    //2. 向下取整
    console.log(Math.floor(n));//1234
    //3. 向上取整
    console.log(Math.ceil(n));//1235
    //4. 产生随机数
    console.log(Math.random());//随机产生 [0,1) 小数
    //5. 产生 [1,10]的随机整数
    //[0,1) *10 --> [0,10) +1 --> [1,11) floor -- [1,10]
    console.log(Math.floor(Math.random() * 10 + 1));//产生 [1,10]的随机整数
</script>
```

```
</body>
</html>
```

4.5-全局函数




```
<!DOCTYPE html>
<html lang="zh">
<head>
    <meta charset="UTF-8">
    <title>全局函数</title>
</head>
<body>
<!--
全局函数
    1. 字符串转为数字
        parseInt();//字符转为整数数字,从左到右遇到非数字停止
        parseFloat();//字符转为小数数字,从左到右遇到非数字停止
        isNaN();//判断非数字
    2. 对数据进行加密
        encodeURI() 把字符串编码为 URI。
    3. 对加密数据进行解密
        decodeURI() 解码某个编码的 URI。
    4. 把字符串当做js表达式来执行
        eval() 计算 JavaScript 字符串，并把它作为脚本代码来执行
-->
<script >
    //1. 字符串转为数字
    console.log(parseInt("123.456"));//123
    console.log(parseInt("12abc3.456"));//12
```

```
    console.log(parseFloat("123.456"));//123.456
    console.log(parseFloat("123.45abc6"));//123.45
    console.log(parseInt("abc123"));//NaN : not a number 不是一个数字
    console.log(isNaN("abc123"));//true
    //2. 对数据进行编码
    let name = "明下午两点偷袭珍珠";
    let encodeName = encodeURI(name);//编码

 console.log(encodeName);//%E6%98%8E%E4%B8%8B%E5%8D%88%E4%B8%A4%E7%82%B9%E5%81%B7%E8%A2%AD%E7%8F%8D%E7%8F%A0

    //3. 对数据进行解码
    let resultStr = decodeURI(encodeName);//解码
    console.log(resultStr);//明下午两点偷袭珍珠
    //4. 把字符串当做js表达式来执行
    let str = "1+2*3-4/2";
    let res = eval(str);//把字符串当做js表达式来执行
    console.log(res);//5


</script>
</body>
</html>
```