JavaScript 基础

第二课

> 学习目标：
>
> 1. JavaScript中的操作符
> 2. JavaScript流程控制

表达式和语句

表达式

> 一个表达式可以产生一个值，有可能是运算、函数调用、字面量等等。表达式可以放在任何需要的地方。
>
> 比如： 5+6

语句

> 语句可以理解为一个行为，循环语句和判断语句就是最经典的语句。一个程序是由很多个语句组成。一条语句，一般以;结束。

操作符

操作符，又叫做运算符 operator

表达式 一般是由 **操作符** 和 **操作数**组成 (常见的表达式)

算术运算符

数学里边的加减乘除

```
// + - * / %
// 跟数学中的是一样的
var x = 5;
var y = 10;
console.log(x + y);
console.log(x - y);
console.log(x * y);
console.log(x / y);
console.log(x % y);

// 两个比较特殊的
console.log(x/0); // Infinity
console.log(x % 0 ); // NaN
```

一元运算符

技术交流QQ：663849976
Web前端交流群：777638970

> 只带一个操作数的 运算符 叫 一元运算符
>
> 能带两个 操作数的 运算符 叫 二元运算符，所以前面所学的 都是 二元运算符。

```
// ++ 自身+1
// -- 自身-1
var num = 5;
console.log(++num);  // 6
console.log(--num);  // 5
```

> `++num` 首先它是一个表达式，既然是表达式，就有 返回值。
>
> 前置和后置的区别就在于这个返回值

- 前置

> 先让操作数 自身+1（-1），然后再返回运算后的结果。

```
var num  = 5;
console.log(++num);// 6;
console.log(num); // 6
var num1 = 7;
console.log(num + ++num1);//14   6+8
console.log(num1);// 8

console.log(--num1);// 7
```

- 后置

> 先返回操作数的结果，然后再对 操作数 自身+1(-1)

```
var num = 5;
console.log(num++);// 5
console.log(num);//6
var num1  = 7;
console.log(num + num1++); // 6+7 13
console.log(num1);// 8
```

- 巩固练习

```
var a = 1;var b= ++a + ++a;console.log(b);
var a = 1;var b= a++ + ++a;console.log(b);
var a = 1;var b = a++ + a++;console.log(b);
var a = 1;var b = ++a + a++;console.log(b);
```

逻辑运算符

> 逻辑运算符又叫 **布尔运算符**。

- &&(逻辑与)： 两个操作数同时为`true`，结果为`true`，否则为`false`。

技术交流QQ：663849976
Web前端交流群：777638970

- ||(逻辑或)：两个操作数有一个为`true`，结果为`true`，否则为`false`。
- !(逻辑非)：取反

```
// 逻辑运算符  与&&    或||    非!
var a = true;
var b = false;
console.log(a && b);
console.log(a || b);
console.log(!a)
```

> 什么时候用到逻辑运算符?
>
> 将来 需要判断 或者 循环的时候，即将使用大量的逻辑运算。

布尔类型的隐式转换

```
var num = 123;
console.log(Boolean(num));// 强制转换

// 隐式转换
// 转换成 false的5中情况  0,'',NaN,undefined,null
if(num){
    console.log('转换成功');
}
var msg;
if(msg){
    console.log('true');
}else{
    console.log('false');
}

// 利用取非进行隐式转换
var str = '哈哈';
var isOk = !!str; //  true，取反 为 false，再取反
console.log(isOk);
```

关系运算符

> 关系运算符又叫 **比较运算符**
>
> 用于比较 两个数的大小 或者 异同 返回值 是*布尔类型*
>
> > 、<、 >=、<= 、== 、!= 、=== 、!==

```
var a = 10;
var b = 5;
var c = '10';

// 比较两个数的大小，返回布尔值
console.log(a>b);
console.log(a<b);
```

技术交流QQ：663849976
Web前端交流群：777638970

```
8
9  // 比较相等的时候，着重讲解
10 // == !=
11 console.log(a==b);
12 console.log(a!=b);
13 // 判断变量的值是否相等
14 console.log(a==c);
15 // === !==
16 console.log(a===b);
17 console.log(a!==b);
18 // 只有 两者的 值 和 数据类型 都相同的时候，才返回true
19 console.log(a===c);
```

赋值运算符

> = 、 += 、 -= 、 *= 、 /= 、 %=

```
1  var num = 5;
2  var num1 = num * 5;
3  // 简化语法
4  num1++; // 自增1
5  num1 = num1 + 5; // 自增5
6  // 简化语法
7  num1 += 5; //  等价于上面
8  // 同理
9  num1 -= 5;
10 num1 *= 5;
11 num1 /= 5;
12 num1 %= 5;
```

运算符的优先级

> 运算符的优先级
>
> 1. () 优先级最高
> 2. 一元运算符 ++ -- !
> 3. 算术运算符 先 * / % 后 + -
> 4. 关系运算符 > >= < <=
> 5. 关系运算符 == != === !==
> 6. 逻辑运算符 先 && 后 ||
> 7. 赋值运算符

```
1  // 练习1
2  4 >= 6 || '人' != '阿凡达' && !(12 * 2 == 144) && true;
3  (4 >= 6) || ('人' != '阿凡达') && (!(12 * 2 == 144)) && true;
4  (4 >= 6) || ('人' != '阿凡达') && (!(24 == 144)) && true;
5  (4 >= 6) || ('人' != '阿凡达') && (!false) && true;
6  (4 >= 6) || ('人' != '阿凡达') && true && true;
7  false || true && true && true;
8  false || true;
9  true;
10
```

技术交流QQ：663849976
Web前端交流群：777638970

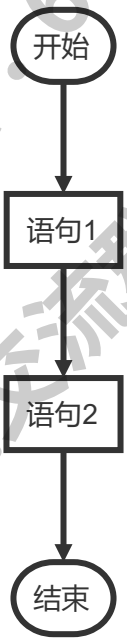
```
// 练习2
var num = 10;
5 == num / 2 && (2 + 2 * num).toString() === '22';
(5 == num / 2) && ((2 + 2 * num).toString() === '22');
(5 == 5) && ((2 + 20).toString() === '22');
true && (22.toString() === '22');
true && ('22' === '22');
true && true;
true;
```

流程控制

> 流程控制 是 编程语言 与 其他语言区分的一个标识，
>
> 流程控制指的是，我们写的代码是如何去执行的。

顺序结构

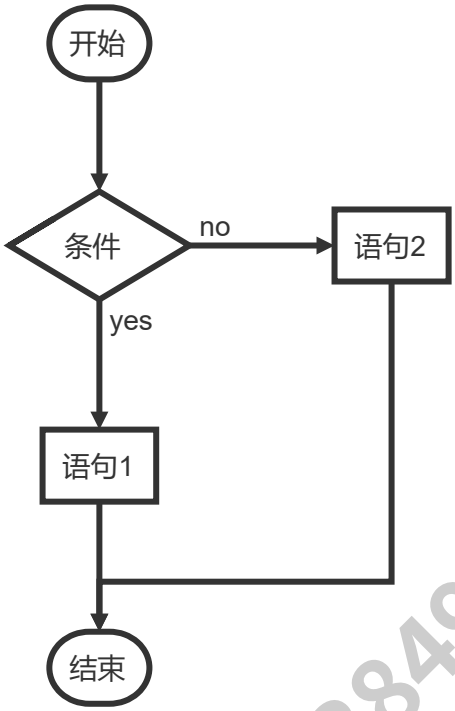
> 代码从上到下依次执行，就是顺序结构。
>
> **程序默认就是从上倒下顺序执行的。**



分支结构

> 根据不同的情况，执行不同的代码
>
> 又叫*选择结构*

技术交流QQ：663849976
Web前端交流群：777638970



if语句

语法：

```
if(/* 条件表达式 */){
    // 执行语句
}


if(/* 条件表达式 */){
    // 成立执行语句
 }else{
    // 否则执行语句
}


if(/* 条件表达式1 */){
    // 成立执行语句
}else if(/* 条件表达式2 */){
    // 成立执行语句
}else if(/* 条件表达式3 */){
    // 成立执行语句
}else{
    // 否则执行语句
}
```

案例：

> 1. 求两个数的最大数
> 2. 判断一个数是奇数还是偶数
> 3. 分数转换，把百分制转换成ABCDE A(90-100) B(80-90) C(70-80) D(60-70) E(<60)

技术交流QQ：663849976
Web前端交流群：777638970

三元运算符

> 分支结构的另类写法
>
> 带有 三个 操作数的运算符
>
> 表达式1? 表达式2 : 表达式3;
>
> 表达式1： 布尔类型的表达式，它总会返回一个布尔类型的值
>
> 当表达式1成立 ，则返回表达式2的值，当表达式1不成立，则返回表达式3的值；
>
> 是对 if else语法的一种简化

```
var num1 = 3;
var num2 = 6;
// 表达式1 ? 表达式2 : 表达式3
console.log(num1>num2?num1:num2);


var age = 17;
console.log(age>=18 ? '成年' : '未成年');
```

switch语句

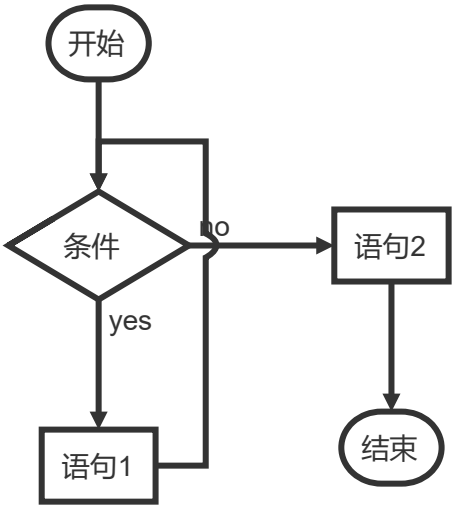
> 也常用语条件判断，用于多个分支。
>
> 只能做 值相等判断，而if可以做某个范围判断

```
switch(expression){
    case 常量1:
        语句;
        break;
    case 常量2:
        语句;
        break;
    case 常量3:
        语句;
        break;
    ...
    case 常量n:
        语句;
        break;
    default:
        语句;
}
```

循环结构

> 重复做一件事。
>
> 在JavaScript中，循环语句有三种，while，do...while，for。

技术交流QQ：663849976
Web前端交流群：777638970



while语句

基本语法：

```
// 当循环条件为true的时候，执行循环体。
// 当循环条件为false的时候，结束循环。

while(循环条件){
    // 循环体
}
```

案例：

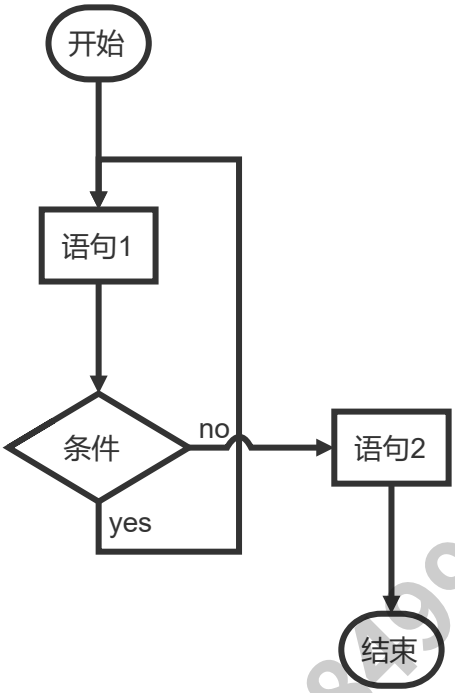
```
// 当循环条件永远是 true的时候，那么就会出现死循环
var i = 1;
while(i<=100){
    // 循环体
    console.log(i);
    i++;
}

// 计算1-100之间的累加和
var i = 1;
var sum = 0;
while(i <= 100){
    sum += i;
    i++;
}
console.log(sum);
```

do-while语句

> do...while和while循环非常像，二者经常可以相互替换，但是do...while的特点是，
>
> 不管条件成立与不成立，都会先执行一次循环体。

技术交流QQ：663849976
Web前端交流群：777638970




```
do{
    // 循环体
}while(循环条件)
```

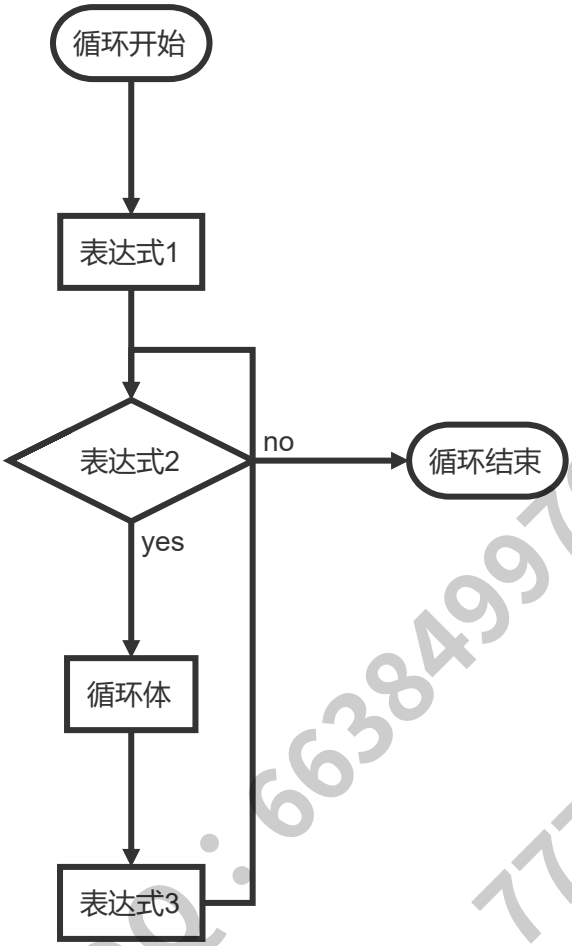
for循环

> while 和do...while一般用于解决无法确认次数的循环。for循环一般在循环次数确定的时候比较方便

```
// for循环的表达式之间用;隔开，千万不能写成，
for(初始化表达式1;条件表达式2;自增表达式3){
    // 循环体 4
}
// 执行顺序
// 1243->243->243直到循环条件变成false
```

> 1. 初始化表达式
> 2. 判断表达式
> 3. 自增表达式
> 4. 循环体

技术交流QQ：663899976
Web前端交流群：777638970



技术交流QQ：663849976
Web前端交流群：777638970

continue和break

> break：立即跳出整个循环，即结束循环。开始执行后边的内容（直接跳到大括号后面）
>
> continue：立即跳出当前循环，继续下一次循环（跳转到i++的地方）

```
// 求 50-200之间 第一个能被7整除的数
for(var i = 50; i<=200; i++ ){
    if(i % 7 === 0){
        console.log(i);
        break;
    }
}


// 求1-100之间的累加值，但是要求 跳过能所有 个位 为 3的数。
var sum  = 0;
for(var i = 0; i<100; i++){
    if( i % 10 === 3){
        console.log(i);
        continue;
    }
    sum += i;
```

```
}
console.log(sum);
```

调试

> debug，调试的目的是确定错误的原因和位置，并解决错误。

- 程序中的错误
  - 语法错误

    浏览器中的console会报错，能快速定位到某个文件某一行
  - 逻辑错误

    程序执行的结果，跟预期的结果不一致，需要自己去分析那一行写错了。
- 简单调试方式
  - alert()
  - console.log()
- 断点调试

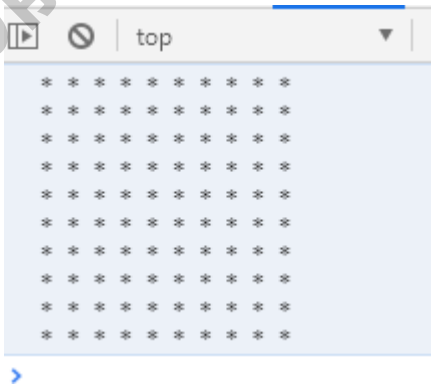
  > 断点调试是指自己在程序中某一行设置一个断点，调试的时候，程序运行到这一步就会停住，然后可以一步一步往下调试。调试过程中，可以看到各个变量当前的值，出错的话，调试到出错的代码行就会显示错误，停止运行。

作业

1. 判断一个人的年龄 是否满18岁 【if语句，三元表达式】
2. 判断一个年份 是 闰年还是平年（闰年：能被4整除，但不能被100整除的年份，或者能被400整除的年份）【if】
3. 打印100以内7的倍数【while、for】
4. 打印100以内 所有的偶数【while、for】
5. 打印100以内所有的奇数和【while、for】
6. 使用do...while，输入询问“我爱你，嫁给我吧？”，选择“你喜欢我吗？(y/n)”

如果输入y则打印，“我们形影不离”，如果输入n，继续询问。

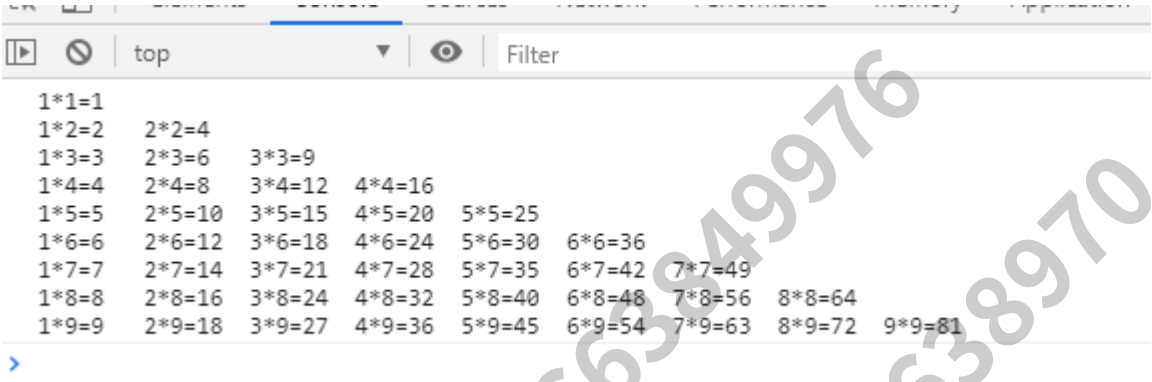
7. 打印1-100之间所有数的平均值
8. 在控制台 输出 一个 10*10 的 * 形状



9. 在控制台输出一个 三角

技术交流QQ：663849976
Web前端交流群：777638970

10. 在控制台 输出 99乘法表



11. 本金10000元存入银行，年利率是 千分之三，每过1年，将本金和利息相加作为新的本金。计算5年后，获得的本金是多少?
12. 有一个人想知道，一年之内一对兔子能繁殖多少对。于是就筑了一道围墙，把一对兔子关在里边。已知一对兔子每月可以生一对小兔子，而一对小兔子出生后第三个月开始每月生一对小兔子。假如1年内没有发生死亡现象，那么1年（12个月）时间，最后剩多少对兔子。

技术交流QQ：663849976
Web前端交流群：777638970