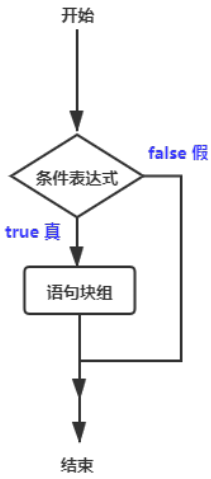
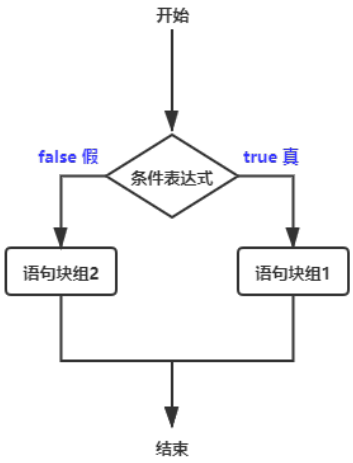
6. JavaScript流程控制

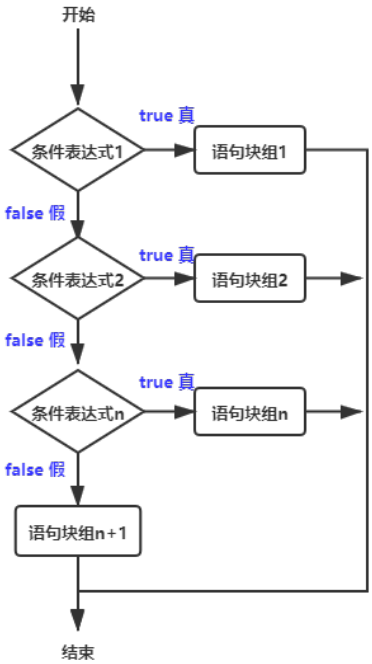
- 任何编程语言都是由一系列语句构成的。
- 一条语句可以是一个赋值语句，一个函数调用，一个循环，甚至一个什么也不做的（空语句）条件语句。
- 在任何一门程序设计语言中，都需要支持满足程序结构化所需要的三种基本结构:
  - 顺序结构
  - 分支结构（选择结构）
  - 循环结构
- 顺序结构：在程序结构中，最基本的就是顺序结构。程序会按照自上而下的顺序执行。由于结构简单所以这里我就不多介绍

1. 分支结构（条件语句）：

- 在`JavaScript`中，我们可使用以下分支语句:
  - `if` 语句 - 只有当指定条件为 `true` 时，使用该语句来执行代码
  - `if...else` 语句 - 当条件为 `true` 时执行代码，当条件为 `false` 时执行其他代码
  - `if...else if....else` 语句 - 使用该语句来选择多个代码块之一来执行
  - `switch...case` 语句 - 使用该语句来选择多个代码块之一来执行







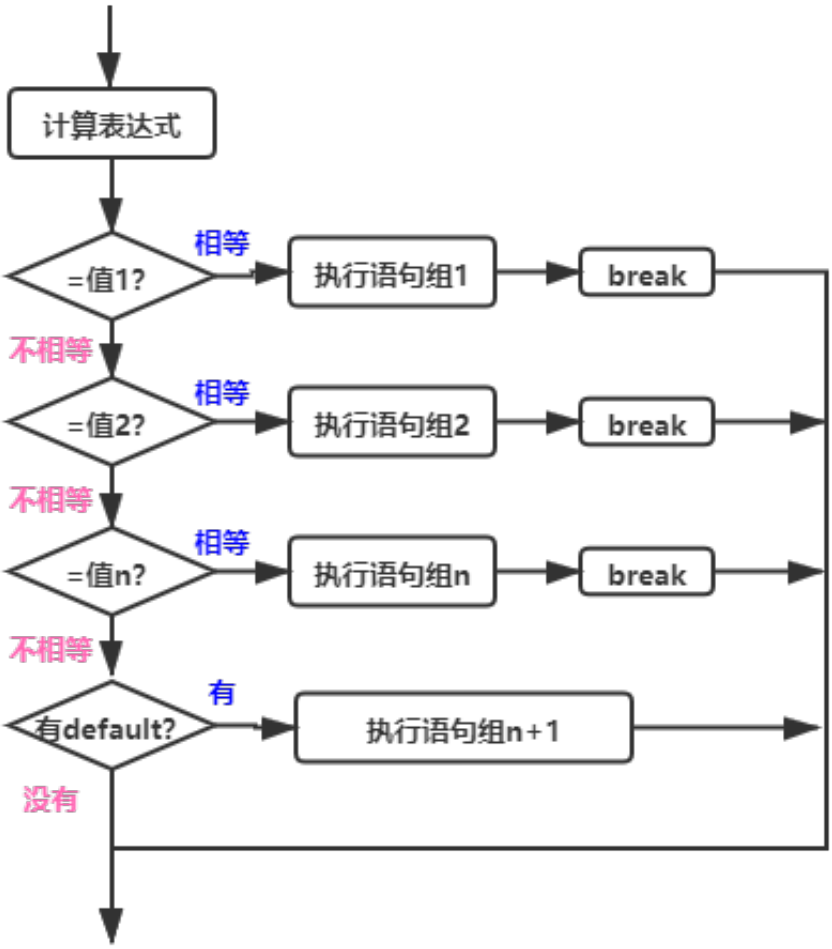
- if .. else 分支判断示例:

```
var grade = 70;
  //单一分支结构
  if(grade>=60){
```

```
    console.log("成绩合格！");
}

//双分支结构
if(grade>=60){
    console.log("成绩合格！");
}else{
  console.log("成绩不及格！");
}


//多分支结构
if(grade>=90){
    console.log("成绩优秀！");
}else if(grade >= 75 ){
  console.log("成绩良好！");
}else if(grade >= 60){
  console.log("你的成绩合格！");
}else{
  console.log("成绩不及格！");
}
```



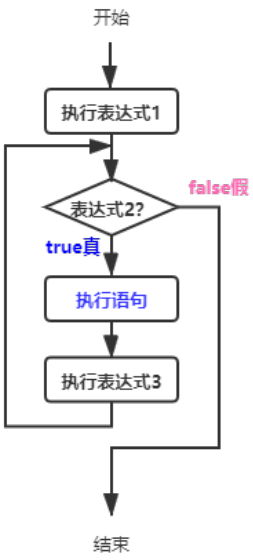
- switch 分支判断示例:

```
//switch实现多分枝判断
//获取今天星期几的数字
var day=new Date().getDay();
switch (day)
{
  case 0:
    x="星期日";
    break;
  case 1:
    x="星期一";
    break;
  case 2:
    x="星期二";
    break;
  case 3:
    x="星期三";
    break;
  case 4:
    x="星期四";
    break;
  case 5:
```

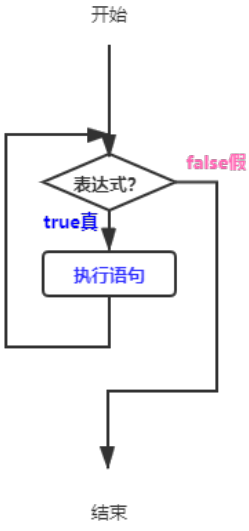
```
      x="星期五";
      break;
    case 6:
      x="星期六";
      break;
    default:
      x="无效的星期信息！";
  }
  console.log(x);
```

2. 循环结构：

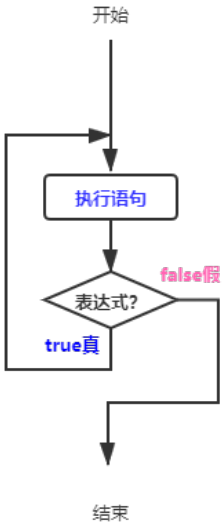
- JavaScript 支持不同类型的循环：
  - `for` - 循环代码块一定的次数
  - `for/in` - 循环遍历对象的属性
  - `while` - 当指定的条件为 `true` 时循环指定的代码块
  - `do/while` - 同样当指定的条件为 `true` 时循环指定的代码块



for循环语句



while循环语句



do ... while 循环语句

2.1 for 循环：

```
for(语句1; 语句2; 语句3){
  被执行的代码块
}

语句1:  在循环（代码块）开始前执行，常用于初始化循环条件
语句2:  定义运行循环（代码块）的条件，用于判断何时结束循环
语句3:  在循环（代码块）已被执行之后执行，常用于递增或递减来影响语句2的判断，直至结束循环
```

- 示例代码：

```
//循环输出1～10的值
for(var i=1;i<=10;i++){
  console.log(i);
}

//计算1～100的累加值
var sum = 0;
for(var i=1;i<=100;i++){
    sum += i;
}
console.log(sum); //5050
```

2.2 for...in 循环遍历对象：

```
for(属性变量 in 被遍历对象){
  被执行的代码块
}
```

- 参考示例

```
var ob = {"name":"张三","age":22,"sex":"男"};
//遍历对象ob中的每个属性
for(key in ob){
  //输出属性和对应的属性值
  console.log(key+":"+ob[key]);
}
/*
//输出结果：
name:张三
age:22
sex:男
*/
```

2.3 while 循环

- While 循环会在指定条件为真时循环执行代码块。

```
while (条件){
  需要执行的代码
}
```

- 注意：如果您忘记增加条件中所用变量的值，该循环永远不会结束。该可能导致浏览器崩溃。
- 参考代码：

```
//循环输出10～1的值
var i = 10;
while(i>=1){
```

```
    console.log(i);
    i--;
}

//计算1～100的累加值
var sum = 0;
var i   = 0;
while(i<=100){
  sum += i;
  i++;
}
console.log(sum); //5050
```

2.4 do/while 循环

- `do/while` 循环是`while`循环的变体。
- 该循环会执行一次代码块，在检查条件是否为真之前，然后如果条件为真的话，就会重复这个循环。
- 该循环至少会执行一次，即使条件是 false，隐藏代码块会在条件被测试前执行。

```
do{
  需要执行的代码;

}while(条件);
```

- 参考示例：

```
//循环输出1～10的值
var i = 1;
do{
  console.log(i);
    i++;
}while(i<=10);

//计算1～100的累加值
var sum = 0;
var i   = 0;
do{
  sum += i;
  i++;
}while(i<=100);
console.log(sum); //5050
```

2.5. 循环中的`break`和`continue`语句

- break 语句用于跳出循环。
- continue 用于跳过循环中的一个迭代。

- break语句
    - 我们已经在本教程稍早的章节中见到过 break 语句。它用于跳出 switch() 语句。
    - break 语句可用于跳出循环。
    - break 语句跳出循环后，会继续执行该循环之后的代码（如果有的话）：
- continue 语句中断循环中的迭代，如果出现了指定的条件，然后继续循环中的下一个迭代。