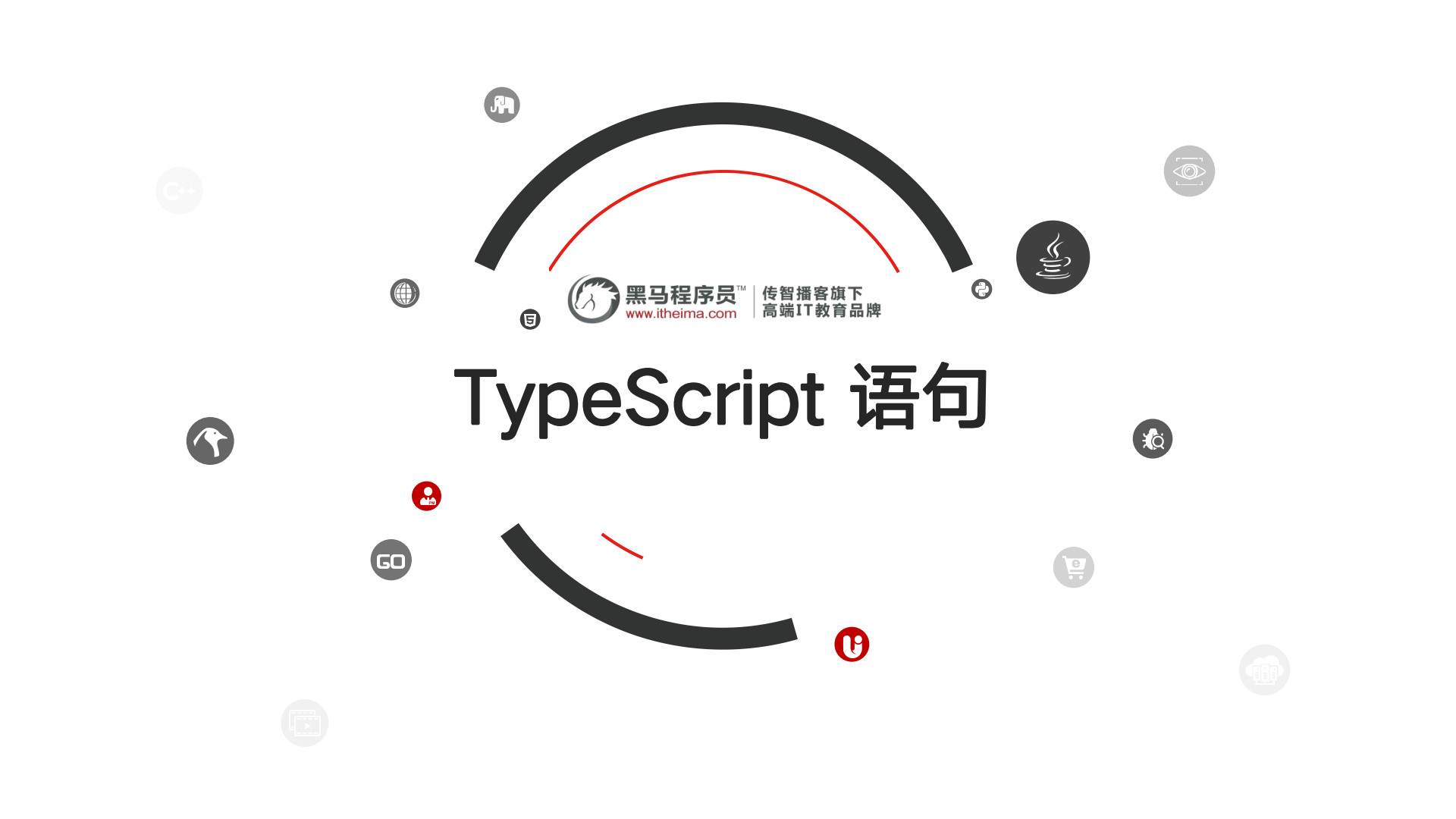

黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
TypeScript 语句

目录 Contents

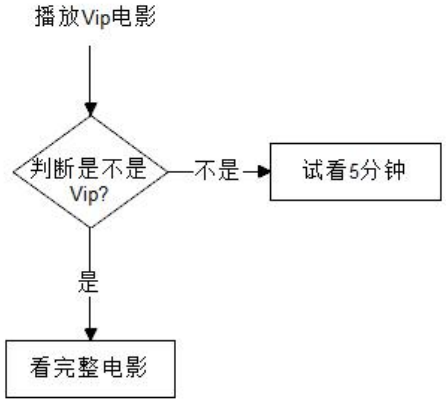
1. 条件语句

1.1 概述

生活中，打开网站看电影：1 免费电影　2 Vip 电影。

播放 Vip 电影时，首先会判断是不是 Vip：

- 如果是 Vip，就可以看完整电影；
- 如果不是 Vip，只能试看5分钟。



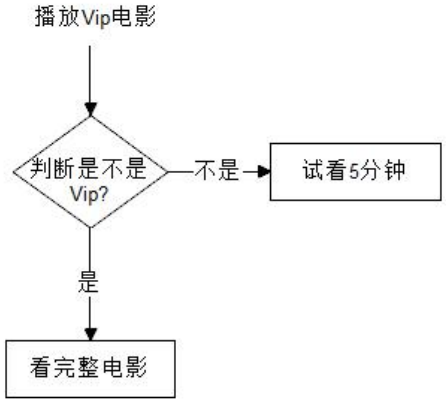
1. 条件语句

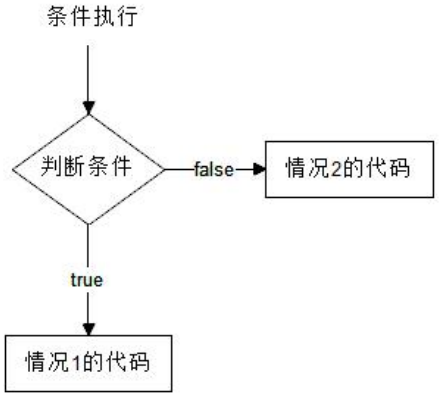
1.1 概述

条件语句：根据判断条件的结果（真或假），来执行不同的代码，从而实现不同功能。

条件执行时，首先判断条件是否满足。

- 如果 **条件满足**，就做某件事情（情况1）
- 如果 **条件不满足**，就做另外一件事情（情况2）





条件语句，也叫分支语句，不同的情况就是不同的分支。

1. 条件语句

1.2 if 语句

在 TypeScript 中 `if` 语句就是实现条件判断的。

- `if` 语句的语法：

```
if (判断条件) {
  条件满足时，要做的事情
}
```

解释：

- 判断条件：布尔类型（true 或 false）。
- 如果 判断条件 为真，就执行 要做的事情；
- 否则，如果判断条件为假，则不执行花括号中的代码。

补充概念说明：语句，是一个完整的句子，用来使某件事情发生（或实现某个功能）。

1.3 else 语句

在 TypeScript 中 `else` 语句必须配合 `if` 语句来使用。

`else` 语句表示：**条件不满足**，要做的事情（`if` 语句的对立面）。

- `else` 语句的语法：

```
if (判断条件) {
  条件满足时，要做的事情
} else {
  条件不满足，要做的事情
}
```

解释：

- 否则，如果 判断条件 为假，就执行 条件不满足时要做的事情。

目录 Contents

2. 三元运算符

三元运算符的作用与 `if...else` 语句类似。

作用：根据判断条件的真假，得到不同的**结果**。

语法：

```
结果 = 判断条件 ? 值1 : 值2
```

解释：

- 如果判断条件为真，结果为 值1；
- 否则，如果判断条件为假，结果为 值2。

注意：得到结果的类型由值1和值2的类型决定（值1和值2的类型相同）。

目录 Contents

3. 循环语句

3.1 概述

生活中，经常重复做某件事情，比如：

1. 上学时作业写 3 遍。
2. 女朋友说：爱我就对我说 100 遍“我爱你”。

需求：在 TS 中，打印 3 遍以下内容：

```
'北冥有鱼，其名为鲲。鲲之大，一锅装不下'
```

在 TypeScript 中，要实现**重复做某件事情**，就需要用到循环语句，来减少重复劳动提升效率。

3. 循环语句

3.2 for 循环

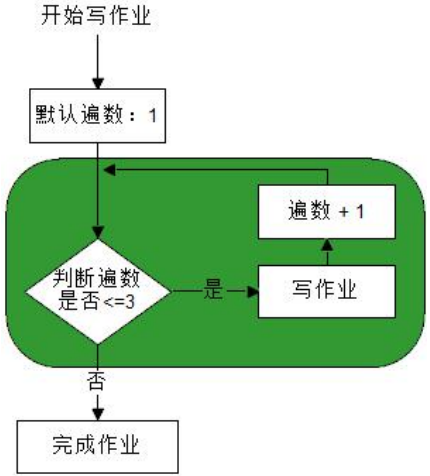
在 TypeScript 中，`for` 循环就是实现**重复做某件事情**的循环语句。

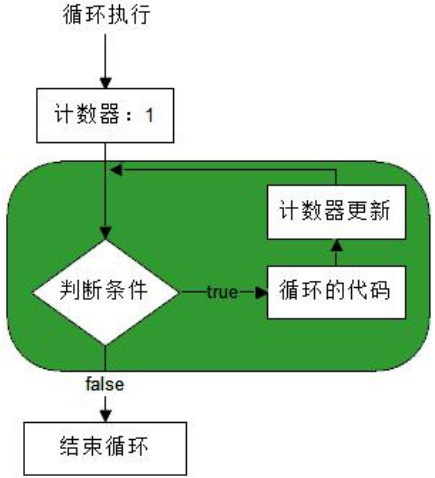
注意：`for` 循环是 TS 基础知识的重难点，语法比较复杂。

3.2 for 循环

上学时作业写 3 遍。先准备，写作业的遍数，默认为：1。

- 第 1 遍：先判断遍数是否 <= 3（是）；写作业；遍数 + 1（准备开始第2遍）。
- 第 2 遍：先判断遍数是否 <= 3（是）；写作业；遍数 + 1（准备开始第3遍）。
- 第 3 遍：先判断遍数是否 <= 3（是）；写作业；遍数 + 1（准备开始第4遍）。
- 第 4 遍：先判断遍数是否 <= 3（否）；结束写作业。



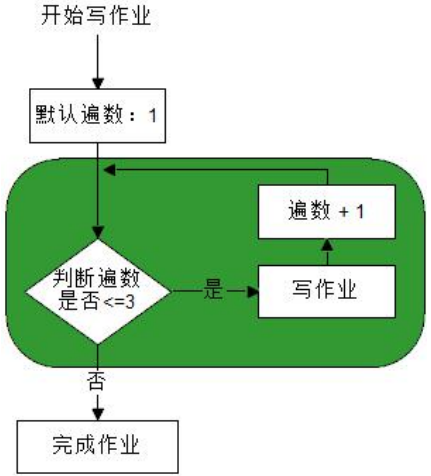


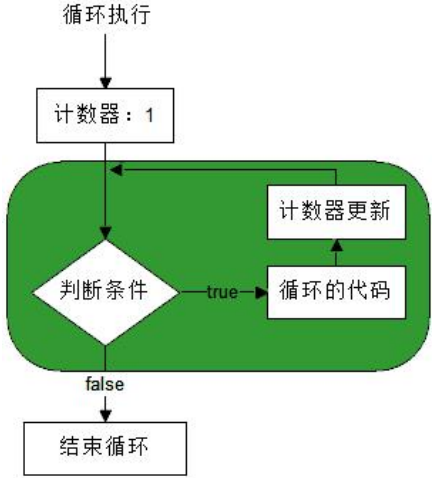
3. 循环语句

3.2 for 循环

`for` 循环的组成：

1. 初始化语句：声明计数器变量用来记录循环次数（**执行一次**）。
2. 判断条件：判断循环次数是否达到目标次数。
3. 计数器更新：完成一次循环让计数器数量加1。
4. 循环体：循环的代码，也就是要重复做的事情。





3. 循环语句

3.3 for 循环的基本使用

- 语法：

```
for (初始化语句; 判断条件; 计数器更新) {
  循环体
}
```

解释：

- 初始化语句：声明计数器变量，记录循环次数。

```
// 作业写 3 遍：
for (let i: number = 1;              )
```

3. 循环语句

3.3 for 循环的基本使用

- 语法：

```
for (初始化语句; 判断条件; 计数器更新) {
  循环体
}
```

解释：

- 初始化语句：声明计数器变量，记录循环次数。
- 判断条件：判断循环次数是否达到目标次数。

```
// 作业写 3 遍：
for (let i: number = 1; i <= 3;     )
```

3.3 for 循环的基本使用

- 语法：

```
for (初始化语句; 判断条件; 计数器更新) {
  循环体
}
```

解释：

- 初始化语句：声明计数器变量，记录循环次数。
- 判断条件：判断循环次数是否达到目标次数。
- 计数器更新：计数器数量加1。

```
// 作业写 3 遍：
for (let i: number = 1; i <= 3; i++)
```

3.3 for 循环的基本使用

- 语法：

```
for (初始化语句; 判断条件; 计数器更新) {
  循环体
}
```

解释：

- 初始化语句：声明计数器变量，记录循环次数。
- 判断条件：判断循环次数是否达到目标次数。
- 计数器更新：计数器数量加1。
- 循环体：重复执行的代码，也就是要重复做的事情。

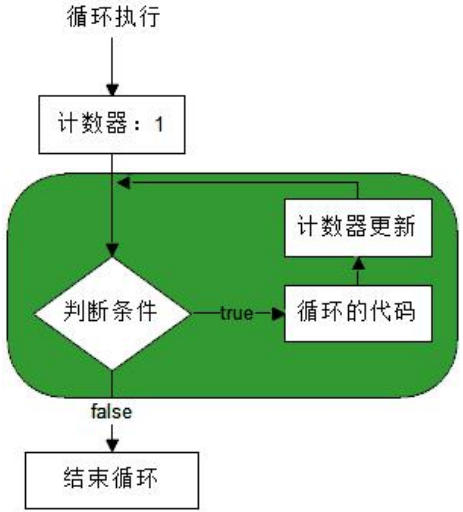
```
// 作业写 3 遍:
for (let i: number = 1; i <= 3; i++) {
  console.log('北冥有鱼，其名为鲲。鲲之大，一锅装不下')
}
```

3.4 for 循环的执行过程

意义：`for` 循环的语法比较复杂，搞明白代码执行顺序，才是真正理解并掌握了 `for` 循环。

1. 初始化语句：只会执行一次。

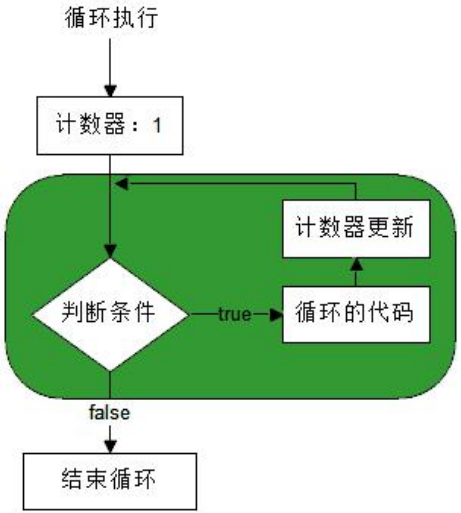
2. 重复执行的部分：判断条件、循环的代码、计数器更新（绿色框框）。



3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



执行过程记录：

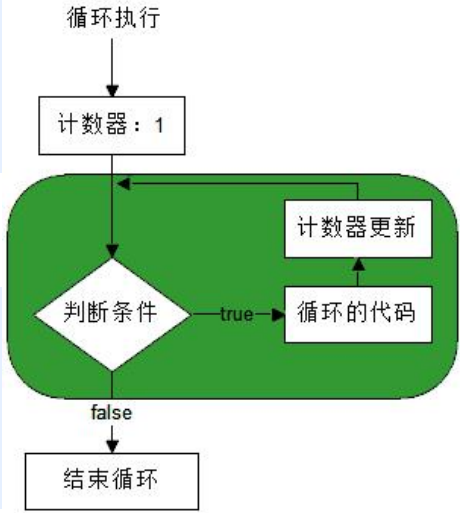
```
// 初始化计数器（i = 1） -- 只执行一次
```

3. 循环语句

3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



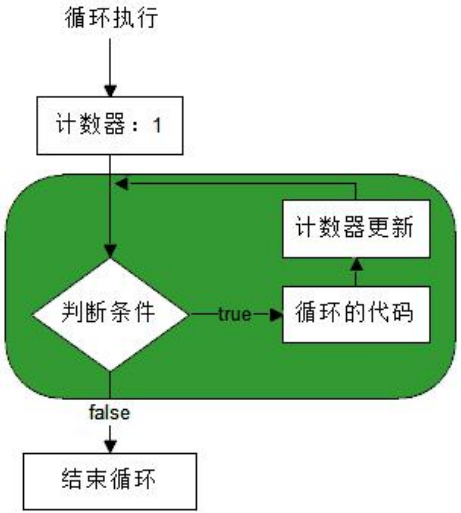
执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3)
```

3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印
```

3. 循环语句

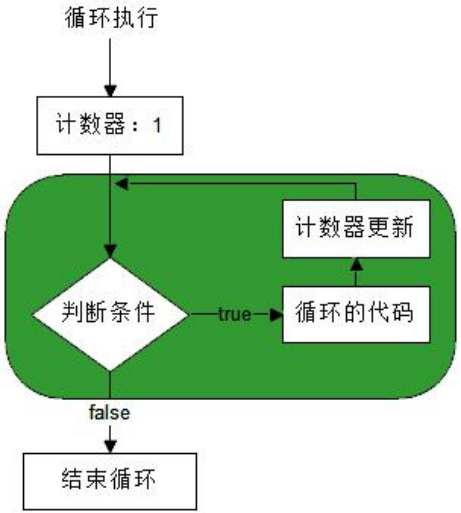
3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```

执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印 c 计数器++ (i变为2) ;
```

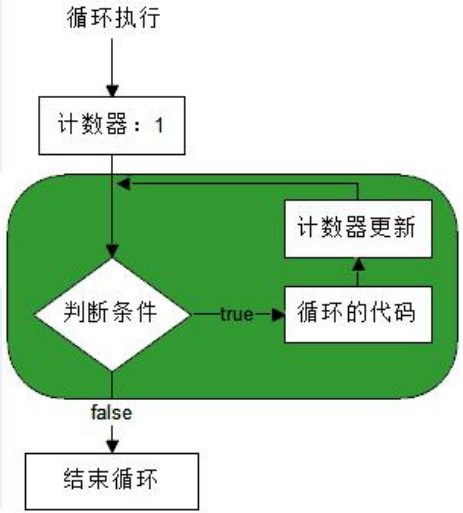


3. 循环语句

3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



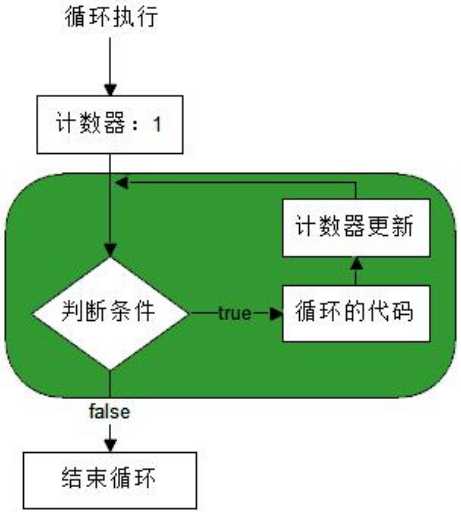
执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印 c 计数器++ (i变为2) ;
// 第 2 次:  a 判断条件 (2 <= 3)
```

3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



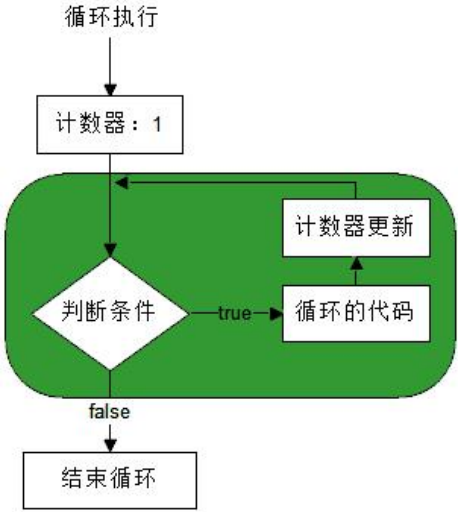
执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印 c 计数器++ (i变为2) ;
// 第 2 次:  a 判断条件 (2 <= 3) b 循环的代码->打印
```

3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



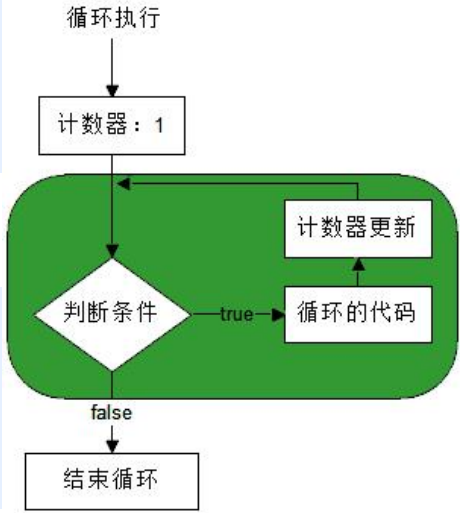
执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印 c 计数器++ (i变为2) ;
// 第 2 次:  a 判断条件 (2 <= 3) b 循环的代码->打印 c 计数器++ (i变为3) ;
```

3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



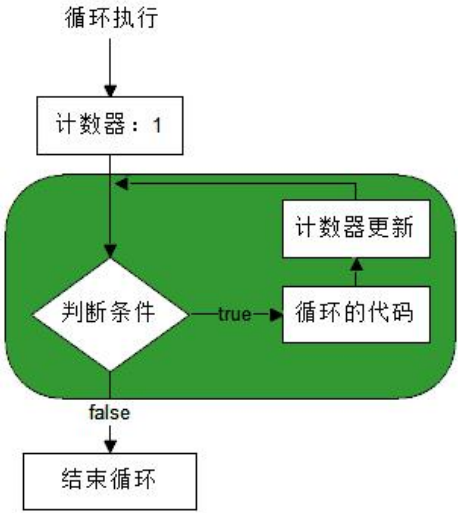
执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印 c 计数器++ (i变为2) ;
// 第 2 次:  a 判断条件 (2 <= 3) b 循环的代码->打印 c 计数器++ (i变为3) ;
// 第 3 次:  a 判断条件 (3 <= 3)
```

3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印 c 计数器++ (i变为2) ;
// 第 2 次:  a 判断条件 (2 <= 3) b 循环的代码->打印 c 计数器++ (i变为3) ;
// 第 3 次:  a 判断条件 (3 <= 3) b 循环的代码->打印
```

3. 循环语句

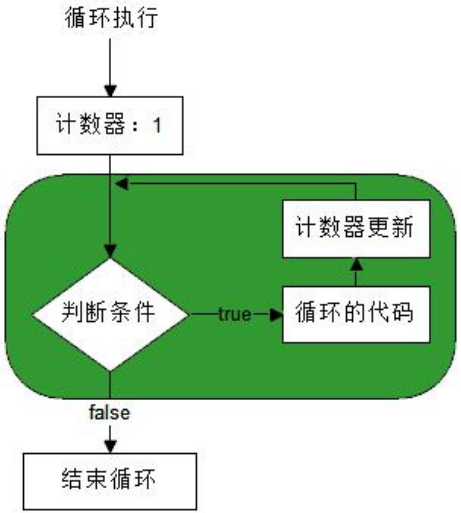
3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```

执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印 c 计数器++ (i变为2) ;
// 第 2 次:  a 判断条件 (2 <= 3) b 循环的代码->打印 c 计数器++ (i变为3) ;
// 第 3 次:  a 判断条件 (3 <= 3) b 循环的代码->打印 c 计数器++ (i变为4) ;
```

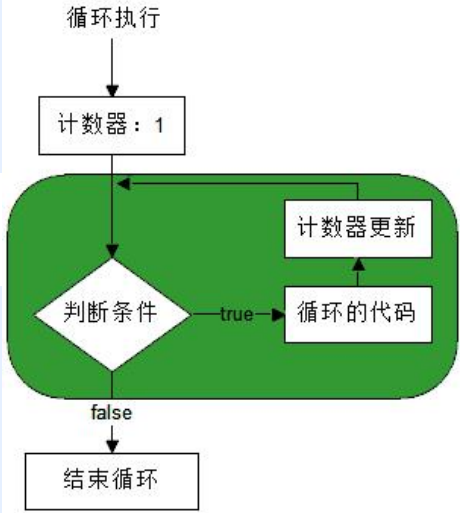


3. 循环语句

3.4 for 循环的执行过程

- 说明：红色表示当前执行的代码。

```
for (let i: number = 1; i <= 3; i++) {
  console.log('...一锅装不下')
}
```



执行过程记录：

```
// 初始化计数器 (i = 1)  -- 只执行一次
// 第 1 次:  a 判断条件 (1 <= 3) b 循环的代码->打印 c 计数器++ (i变为2) ;
// 第 2 次:  a 判断条件 (2 <= 3) b 循环的代码->打印 c 计数器++ (i变为3) ;
// 第 3 次:  a 判断条件 (3 <= 3) b 循环的代码->打印 c 计数器++ (i变为4) ;
// 第 4 次:  a 判断条件 (4 <= 3)，条件不满足，结束循环。
```

3. 循环语句

3.5 断点调试

疑问：老师是如何知道 `for` 循环执行过程的呢？断点调试

借助断点调试，观察代码的执行过程。

断点（Breakpoint）：程序暂停的位置（调试时，程序运行到此处，就会暂停）。

```
1  // 断点调试
2
3  console.log('1 准备开始执行 for 循环')
4
5  for (let i: number = 1; i <= 3; i++) {
6    console.log('...一锅装不下')
7  }
```

3. 循环语句

3.6 break和continue

`break` 和 `continue` 常用在循环语句中，用来改变循环的执行过程。

`for` 循环执行的特点是：连续且不间断。

例子：买了 5 个包子，吃包子。

```
for (let i: number = 1; i <= 5; i++) {
  console.log('正在吃第' + i + '个包子')
}
```

3. 循环语句

3.6 break和continue

`break` 能够让循环提前结束（终止循环）。

例子：买了 5 个包子，吃包子。

场景：吃到（没吃）第3个饱了，剩下的就不吃了。

```
for (let i: number = 1; i <= 5; i++) {



  console.log('正在吃第' + i + '个包子')
}
```

3. 循环语句

3.6 break和continue

`break` 能够让循环提前结束（终止循环）。

例子：买了 5 个包子，吃包子。

场景：吃到（没吃）第3个饱了，剩下的就不吃了。

```
for (let i: number = 1; i <= 5; i++) {
  if (i === 3) {

  }
  console.log('正在吃第' + i + '个包子')
}
```

3. 循环语句

3.6 break和continue

break 能够让循环提前结束（终止循环）。

例子：买了 5 个包子，吃包子。

场景：吃到（没吃）第3个饱了，剩下的就不吃了。

```
for (let i: number = 1; i <= 5; i++) {
  if (i === 3) {
    break
  }
  console.log('正在吃第' + i + '个包子')
}
```

3. 循环语句

3.6 break和continue

`continue` 能够让循环间断执行（跳过本次循环，继续下一次循环）。

例子：买了 5 个包子，吃包子。

场景：吃到第3个有虫子，这个就不再吃了，但没吃饱，继续吃下一个。

```
for (let i: number = 1; i <= 5; i++) {



  console.log('正在吃第' + i + '个包子')
}
```

3. 循环语句

3.6 break和continue

continue 能够让循环间断执行（跳过本次循环，继续下一次循环）。

例子：买了 5 个包子，吃包子。

场景：吃到第3个有虫子，这个就不再吃了，但没吃饱，继续吃下一个。

```
for (let i: number = 1; i <= 5; i++) {
  if (i === 3) {

  }
  console.log('正在吃第' + i + '个包子')
}
```

3.6 break和continue

continue 能够让循环间断执行（跳过本次循环，继续下一次循环）。

例子：买了 5 个包子，吃包子。

场景：吃到第3个有虫子，这个就不再吃了，但没吃饱，继续吃下一个。

```
for (let i: number = 1; i <= 5; i++) {
  if (i === 3) {
    continue
  }
  console.log('正在吃第' + i + '个包子')
}
```

传智播客旗下高端IT教育品牌