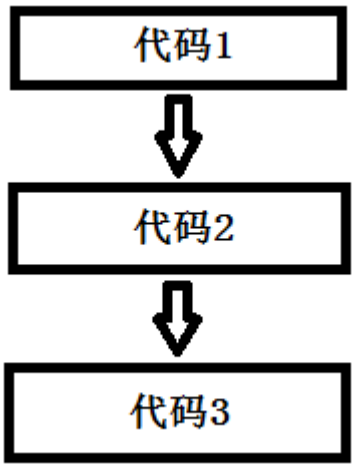
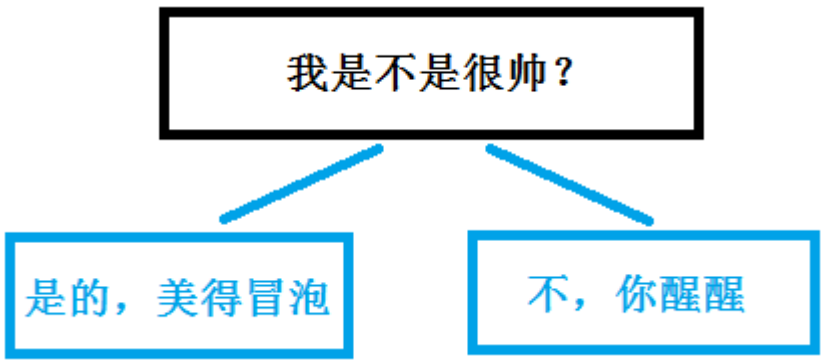
程序的三大结构

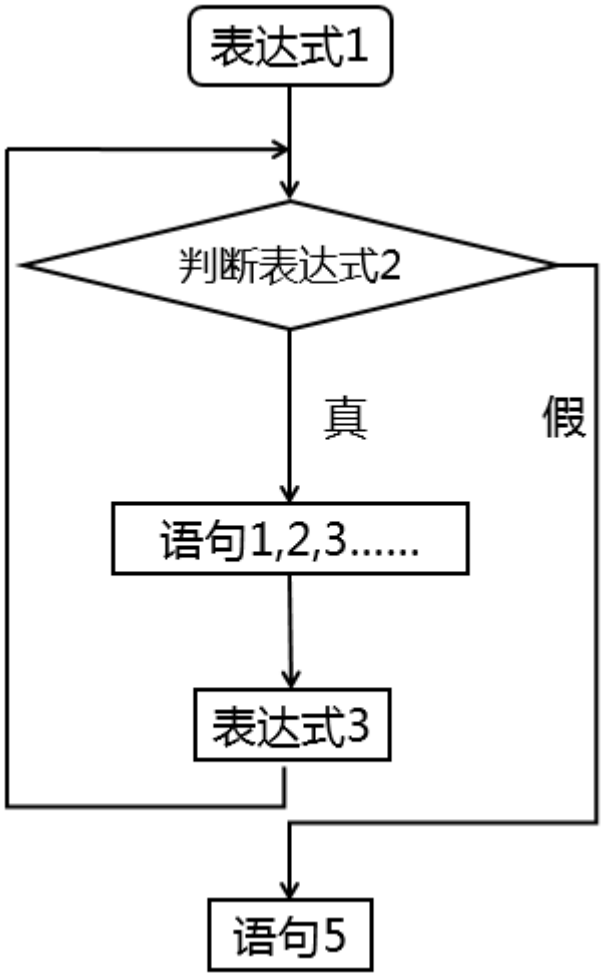
顺序结构



选择结构



循环结构



分支结构

- 我们的 js 代码都是顺序执行的（从上到下）
- 逻辑分支就是根据我们设定好的条件来决定要不要执行某些代码

IF 条件分支结构

if 语句

- 通过一个 if 语句来决定代码执行与否
- 语法：`if（条件）{ 要执行的代码 }`
- 通过 `()` 里面的条件是否成立来决定 `{}` 里面的代码是否执行

```
// 条件为 true 的时候执行 {} 里面的代码
if (true) {
  alert('因为条件是 true，我会执行')
}

// 条件为 false 的时候不执行 {} 里面的代码
if (false) {
    alert('因为条件是 false，我不会执行')
}
```

if else 语句

- 通过 if 条件来决定，执行哪一个 `{}` 里面的代码
- 语法：`if (条件) { 条件为 true 的时候执行 } else { 条件为 false 的时候执行 }`
- 两个 `{}` 内的代码一定有一个会执行

```
// 条件为 true 的时候，会执行 if 后面的 {}
if (true) {
  alert('因为条件是 true，我会执行')
} else {
  alert('因为条件是 true，我不会执行')
}

// 条件为 false 的时候，会执行 else 后面的 {}
if (false) {
  alert('因为条件为 false，我不会执行')
} else {
  alert('因为条件为 false，我会执行')
}
```

if else if ... 语句

- 可以通过 if 和 else if 来设置多个条件进行判断
- 语法：`if (条件1) { 条件1为 true 的时候执行 } else if (条件2) { 条件2为 true 的时候执行 }`
- 会从头开始依次判断条件
  - 如果第一个条件为 true 了，那么就会执行后面的 `{}` 里面的内容
  - 如果第一个条件为 false，那么就会判断第二个条件，依次类推
- 多个 `{}`，只会有一个被执行，一旦有一个条件为 true 了，后面的就不在判断了

```
// 第一个条件为 true，第二个条件为 false，最终会打印“我是代码段1”
if (true) {
  alert('我是代码段1')
} else if (false) {
```

```
    alert('我是代码段2')
}

// 第一个条件为 true，第二个条件为 true，最终会打印“我是代码段1”
// 因为只要前面有一个条件满足了，就不会继续判断了
if (true) {
  alert('我是代码段1')
} else if (true) {
  alert('我是代码段2')
}

// 第一个条件为 false，第二个条件为 true，最终会打印“我是代码段2”
// 只有前一个条件为 false 的时候才会继续向后判断
if (false) {
  alert('我是代码段1')
} else if (true) {
  alert('我是代码段2')
}

// 第一个条件为 false，第二个条件为 false，最终什么也不会发生
// 因为当所有条件都为 false 的时候，两个 {} 里面的代码都不会执行
if (false) {
  alert('我是代码段1')
} else if (false) {
  alert('我是代码段2')
}
```

if else if ... else 语句

- 和之前的 `if else if ...` 基本一致，只不过是在所有条件都不满足的时候，执行最后 else 后面的 `{}`

```
// 第一个条件为 false，第二个条件为 false，最终会打印“我是代码段3”
// 只有前面所有的条件都不满足的时候会执行 else 后面的 {} 里面的代码
// 只要前面有一个条件满足了，那么后面的就都不会执行了
if (false) {
  alert('我是代码段1')
} else if (false) {
  alert('我是代码段2')
} else {
  alert('我是代码段3')
}
```

prompt(str1，str2) 弹出可输入的对话框

str1: 提示要显示在消息对话框中的文本

str2：文本框中的内容

返回值:

1. 点击确定按钮，文本框中的内容将作为函数的返回值

2. 点击取消按钮，将返回null

SWITCH 条件分支结构

- 也是条件判断语句的一种
- 是对于某一个变量的判断
- 语法：

```
switch (n) {
  case 情况1:
    情况1要执行的代码
    break
  case 情况2:
    情况2要执行的代码
    break
  case 情况3:
    情况3要执行的代码
    break
  default:
    上述情况都不满足的时候执行的代码
}

工作原理：
首先设置表达式 n（通常是一个变量）。
随后表达式的值会与结构中的每个 case 的值做比较。
如果存在匹配，则与该 case 关联的代码块会被执行。
使用 break 来阻止代码自动地向下一个 case 运行。
注：break关键字会导致代码执行流跳出switch语句。
```

- 例子🌰： 根据变量给出的数字显示是星期几

```
var week = 1
switch (week) {
  case 1:
    alert('星期一')
    break
  case 2:
    alert('星期二')
    break
  case 3:
    alert('星期三')
    break
  case 4:
    alert('星期四')
    break
  case 5:
    alert('星期五')
    break
  case 6:

    alert('星期六')
```

```
    break
  case 7:
    alert('星期日')
    break
  default:
    alert('请输入一个 1 ~ 7 之间的数字')
}
```

if...else if...else语句与switch case语句的比较：

范围：前者可以比较定值也可以比较范围

后者只能比较定值

效率：前者效率低（每一个表达式都要求值对比）

后者效率高（表达式只需要跟case中的一个匹配就可以）

三元运算（扩展）

- 三元运算，就是用 **两个符号** 组成一个语句
- 三元运算只是对 **if else** 语句的一个简写形式
- 语法：`条件 ? 条件为 true 的时候执行 : 条件为 false 的时候执行`

```
var age = 18;
age >= 18 ? alert('已经成年') : alert('没有成年')
```