一、流程控制

1. 作用

控制代码的执行顺序

2. 分类

1） 顺序结构

从上到下依次执行代码语句

2） 分支/选择结构

1. if语句

- 简单if结构

```
if(条件表达式){
    表达式成立时执行的代码段
}
```

注意 : 除零值以外,其他值都为真， 以下条件为假值false

```
if(0){}
if(0.0){}
if(""){} //空字符串
if(undefined){}
if(NaN){}
if(null){}
```

特殊写法 :
{}可以省略,一旦省略， if语句只控制其后的第一行代码

- if - else结构

```
if(条件表达式){
    //条件成立时执行
}else{
    //条件不成立时选择执行
}
```

- 多重分支结构

```
if(条件1){
  //条件1成立时执行
}else if(条件2){
  //条件2成立时执行
}else if(条件3){
  //条件3成立时执行
}...else{
  //条件不成立时执行
}
```

2. switch语句

- 语法：

```
switch(value){
     case 值1 :
     //value与值1匹配全等时,执行的代码段
     break; //结束匹配
     case 值2 :
     //value与值2匹配全等时,执行的代码段
     break;
     case 值3 :
     //value与值3匹配全等时,执行的代码段
     break;
     default:
     //所有case匹配失败后默认执行的语句
     break;
}
```

- 使用：

```
1. switch语句用于值的匹配,case用于列出所有可能的值;只有switch()表达式的值与case的值匹配全等时,才会执行case对应的代码段
2. break用于结束匹配,不再向后执行;可以省略,break一旦省略,会从当前匹配到的case开始,向后执行所有的代码语句,直至结束或碰到break跳出
3. default用来表示所有case都匹配失败的情况,一般写在末尾,做默认操作
4. 多个case共用代码段
        case 值1:
        case 值2:
        case 值3:
        //以上任意一个值匹配全等都会执行的代码段
```

3. 循环结构

- 作用
  根据条件,重复执行某段代码
- 分类

1. while循环

```
定义循环变量;
  while(循环条件){
   条件满足时执行的代码段
   更新循环变量;
  }
```

2. do-while循环

```
do{
    循环体;
    更新循环变量
}while(循环条件);
```

与while循环的区别 :

- while循环先判断循环条件,条件成立才执行循环体
- do-while循环不管条件是否成立,先执行一次循环体

3. for循环

```
for(定义循环变量;循环条件;更新循环变量){
    循环体;
}
```

循环控制 :

1. break 强制结束循环
2. continue 结束当次循环,开始下一次循环

循环嵌套 :

在循环中嵌套添加其他循环