数组的操作

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

数组的操作

主要针对数组中的数据进行 查找、修改、增加 或 删除

操作	语法
查找	数组名[下标]、数组名.length
修改	数组名[下标] = 新值
增加	数组名.push(数据1, 数据2, ...)、 数组名.unshift(数据1, 数据2, ...)
删除	数组名.pop()、数组名.shift()
任意位置增加或删除	数组名.splice(操作的起始位置, 删除的个数, 新增1, 新增2, ……)

查找 & 修改

查找: 数组名[下标]

修改: 数组名[下标] = 新值

数组长度：数组名.length


```
let names: string[] = ['小明', '小红', '大强', '小飞']

// 1. 查找
console.log('查找姓名', names[0])

// 2. 长度
console.log('数组长度为', names.length)

// 3. 修改
names[1] = 'Jack'
console.log('names数组', names)
```

增加数组元素

往开头加：数组名.unshift(数据1, 数据2, 数据3, ……)

结尾添加：数组名.push(数据1, 数据2, 数据3, ……)

```
let songs: string[] = ['告白气球', '洋葱', '吻别']

// unshift(): 开头新增 (返回操作后数组的长度)
songs.unshift('你是我的眼')

// push(): 结尾新增    (返回操作后数组的长度)
songs.push('光辉岁月', '海阔天空')

console.log('数组songs', songs)
```

删除数组元素

从开头删：数组名.shift()

从结尾删：数组名.pop()

```
let songs: string[] = ['告白气球', '洋葱', '吻别']

// shift(): 开头删除 (返回值: 删除的项)
songs.shift()

// pop(): 结尾删除 (返回值: 删除的项)
songs.pop()

console.log('数组songs', songs)
```

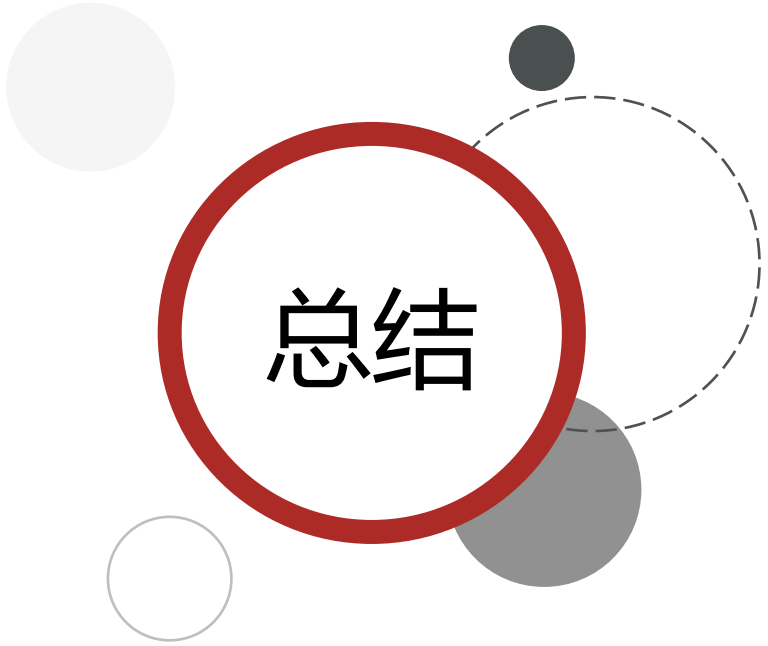
任意位置添加 / 删除数组元素

语法：数组名.splice(起始位置，删除的个数，新增元素1，新增元素2，.....)

```
let songs: string[] = ['告白气球', '洋葱', '吻别']

// 删除下标为2的元素  ['告白气球', '洋葱']
songs.splice(2, 1)


console.log('数组songs', songs)
```



数组的操作？

操作	语法
查找	数组名[下标]、数组名.length
修改	数组名[下标] = 新值
增加	数组名.push(数据1, 数据2, ...) 数组名.unshift(数据1, 数据2, ...)
删除	数组名.pop() 数组名.shift()
任意位置增加或删除	数组名.splice(起始位置, 删除的个数, 新增1, 新增2, ……)

语句

黑马程序员
www.itheima.com

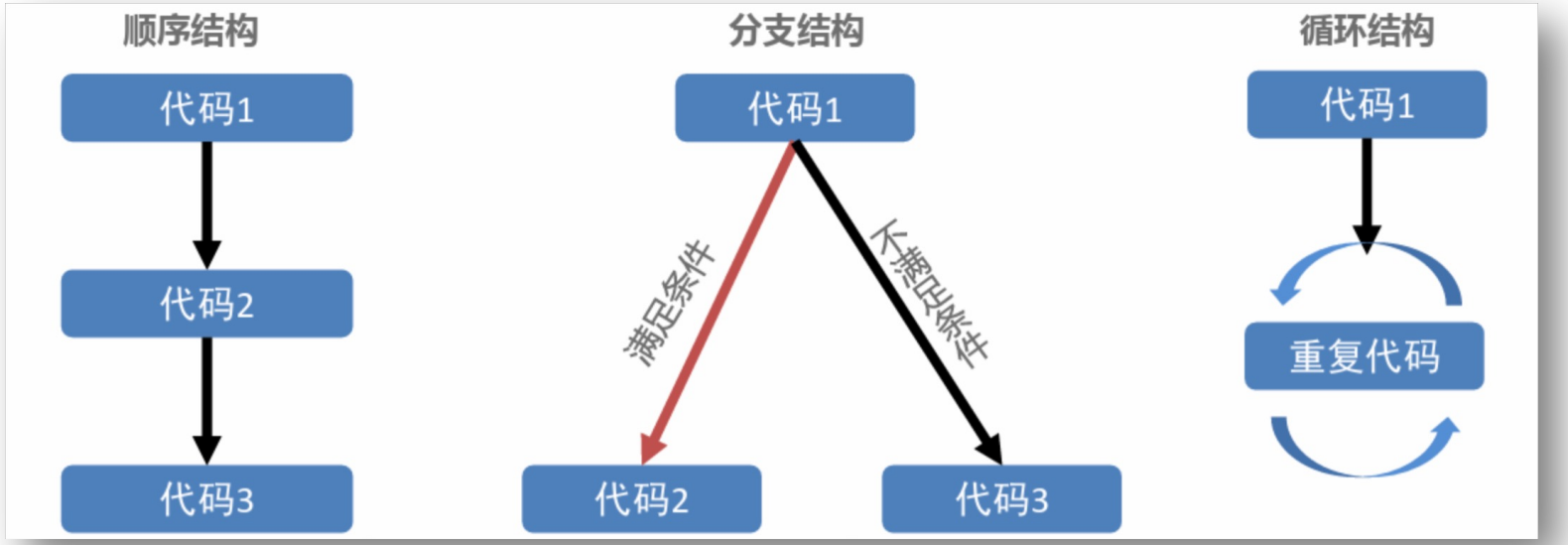
传智教育旗下
高端IT教育品牌

语句概念

语句： 一段可以执行的代码，是一个行为（num = a + b）

表达式： 可以 被求值 的代码，并将其计算出 一个结果 （1 + 1、3 * 5、3 > 2）

语句执行结构：



分支语句

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

if 分支语句

if 分支语句：根据 逻辑条件 不同，执行不同语句。

```
if (逻辑条件) {
  条件成立执行的代码
}
```

单分支语法

```
// 如果分数大于80，才能奖励[周末去游乐园]
let score: number = 82

if (score >= 80) {
  console.log('单分支:', '分数大于80，周末可以去游乐园')
}
```

- 小括号条件结果为 true，则执行大括号里面的代码
- 小括号结果不是布尔类型时，会类型转换为布尔值

if 分支语句

if 分支语句：根据 逻辑条件 不同，执行不同语句。

```
if (逻辑条件) {
  条件成立执行的代码
}
else {
  条件不成立执行的代码
}
```

双分支语法

```
// 如果分数大于80，才能奖励[周末去游乐园]
let score: number = 56

if (score >= 80) {
  console.log('通过', '分数大于80，周末可以去游乐园')
}
else {
  console.log('不通过', '分析问题，写个检讨')
}
```

总结

1. 三种语句执行结构？

   顺序结构、分支结构、循环结构。

2. 分支语句 - if 的语法？

```
if (逻辑条件) {
  条件成立执行的代码
}
```

单分支

```
if (逻辑条件) {
  条件成立执行的代码
}
else {
  条件不成立执行的代码
}
```

双分支

案例 – 购物车数字框

需求：

- 购物车商品 **数量大于1** 可以单击“-”按钮
- 否则 **提示** "最小数量为1，不能再减了"

if 多分支

if 多分支，可以解决多种分支的情况

```
if (条件1) {
  条件1成立执行的代码
}
else if (条件2) {
  条件2成立执行的代码
}
else if (条件3) {
  条件3成立执行的代码
}
else {
  都不成立执行的代码
}
```

```
let score: number = 66

if (score >= 90) {
  console.log('优秀')
} else if (score >= 70) {
  console.log('良好')
} else if (score >= 60) {
  console.log('及格')
} else {
  console.log('不及格')
}
```

if 多分支 – 小作业

需求：根据不同年纪，给出不同的生活建议

- 年龄 < 18: "你还是个孩子，应该多学习。"
- 18 <= 年龄 < 30: "你正年轻，可以多去旅行，拓展视野。"
- 30 <= 年龄 < 50: "你处于事业的黄金期，应该专注于工作和家庭。"
- 50 <= 年龄 < 70: "你已经积累了丰富的人生经验，可以考虑分享给小朋友们。"
- 年龄 >= 70: "你应该享受晚年生活，多和家人在一起。"

switch 分支

switch 分支一般用于精确匹配，不同的值执行不同的代码

```
switch (表达式) {
  case 值1:
    与值1匹配执行的语句
    break
  case 值2:
    与值2匹配执行的语句
    break
  default:
    以上都未成功匹配执行的代码
}
```

```
let fruit: string = '苹果'
// 水果不同，提示不同的水果价格
switch (fruit) {
  case '苹果':
    console.log('苹果:', '2.8元1斤')
    break
  case '鸭梨':
    console.log('鸭梨:', '5.5块1斤')
    break
  case '西瓜':
    console.log('西瓜:', '1.9元1斤')
    break
  default:
    console.log('提示:', '没有您要的水果，请重新输入~')
}
```

注意：如果没有break语句，则会直接执行 switch 中的下一个代码块（无论是否匹配成功）

三元条件表达式

语法：条件？条件成立执行的表达式：条件不成立执行的表达式

```
let num1: number = 5
let num2: number = 10
// 返回较大值
let res: number = num1 > num2 ? num1 : num2
console.log('結果是', res)
```

条件渲染

条件渲染：使用 if、else 和 else if，可基于 不同状态 渲染 对应不同 UI 内容。

```
@State counter: number = 1

build() {
  Column() {
    if (this.counter == 1) {
      Text('1')
    } else if (this.counter == 2){
      Text('2')
    } else {
      Text('any')
    }
  }
}
```

条件渲染案例 – 京东加购

需求：

① 有库存 显示“加购“按钮

② 无库存 显示“查看类似商品”和 提示信息

循环语句

while 语句 和 for 语句

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

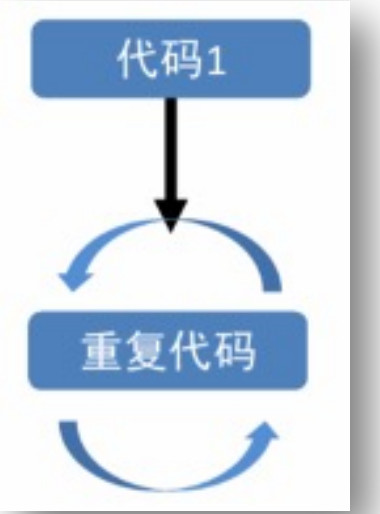
while 语句

作用：重复执行指定的一段代码

```
while (条件) {
  条件成立重复执行的代码
}

/*
while (true) {
  console.log('while', '重复执行的代码')
}
*/

// 指定循环次数
let i: number = 1
while (i < 5) {
  console.log('while~i', '重复执行的代码')
  i++
}
```



Tips：循环三要素

1. 初始值（变量）
2. 循环条件
3. 变化量（变量计数，自增或自减）

while 语句 – 练习

需求1：打印 1-100 的数字

需求2：打印 1-100 中的偶数 （能被 2 整除）

需求3：计算 1-10 内数字的累加和

for 语句

作用：重复执行指定的一段代码

```
for (初始值; 条件; 变化量) {
  重复执行的代码
}
```

```
for (let i: number = 0; i < 5; i++) {
  console.log('for', '重复执行的代码')
}
```

Tips：循环三要素

1. 初始值（变量）
2. 循环条件
3. 变化量（变量计数，自增或自减）

练习：计算 1-10 内数字的累加和

退出循环

作用：满足指定条件，可以退出循环

- break：终止整个循环
- continue： 退出当前一次循环的执行，继续执行下一次循环

```
for (let i: number = 1; i <= 8; i++) {
  if (i == 3) {
    console.log('吃饱了，不吃了')
    // 终止循环
    break
  }
  console.log(`正在吃第${i}个包子`)
}
```

```
for (let i: number = 1; i <= 8; i++) {
  if (i == 3) {
    console.log('这个包子好像坏了，这个不吃了')
    continue
  }
  console.log(`正在吃第${i}个包子`)
}
```

退出循环 – 练习

需求1：打印 1-100 的数字，遇到 7 的倍数跳过

需求2：打印 1-100 中的偶数，遇到 20，后面的就不打印了

遍历数组

遍历：将数组里面的每个数据，按顺序访问一遍

```
let names: string[] = ['小红', '小明', '大强']

for(let i = 0; i < names.length; i++) {
  console.log('名字是', names[i])
}
```

遍历数组 – for ... of

语法：`for (let item of 数组名) {}`

- for ... of : 在 ... 之中 进行循环
- item: 声明的一个变量，用来在循环的时候接收 每一个数组元素

```
let names: string[] = ['小红', '小明', '大强']

for(let i = 0; i < names.length; i++) {
  console.log('名字是', names[i])
}

for (let item of names) {
  console.log('for...of...名字是', item)
}
```

案例

需求1： 求出下列数组元素的 累加和

[22, 3, 44, 55, 80]

需求2：筛选 数组中 大于等于10 的 元素，收集到一个新数组中

[22, 3, 44, 55, 80, 10, 11, 5, -1]

需求3：数组去0，将数组中 不是0 的项收集到一个新数组中

[22, 3, 0, 55, 0, 0, 11, 5, 0]

传智教育旗下高端IT教育品牌