

字符串拼接



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

字符串拼接

作用：把两个或多个字符串，**拼成一个字符串**。（通常拼接的是字符串和变量）

```
'hello' + 'world'    =>    'helloworld'
```

加号的作用：拼接

```
let name: string = '小明'  
console.log('简介信息', '名字是')
```

字符串拼接

作用：把两个或多个字符串，**拼成一个字符串**。（通常拼接的是字符串和变量）

```
'hello' + 'world' => 'helloworld'
```

加号的作用：拼接

```
let name: string = '小明'  
console.log('简介信息', '名字是' + name)
```



简介信息 名字是 **小明**

注意：加法两端只要有**字符串**，就是拼接



总结

1. 字符串拼接的作用？

多个字符串拼成一个整体，拼接 **字符串** 和 **变量**

2. 使用的符号是什么？

+（两端只要有字符串，就是拼接）

```
let name: string = '小美'  
console.log('简介信息', '名字是' + name)
```

模板字符串

模板字符串 `hello`

作用：拼接字符串和变量

优势：更适用于 **多个变量** 的字符串拼接

```
let name: string = '小明'  
let age: number = 18  
console.log('简介信息', `姓名是小明, 今年18岁了`)
```

模板字符串 `hello`

作用：拼接字符串和变量

优势：更适用于 **多个变量** 的字符串拼接

```
let name: string = '小明'
let age: number = 18
console.log('简介信息', `姓名是${name},今年${age}岁了`)
```

适合复杂场景

```
let name: string = '小明'
let age: number = 18
console.log('简介信息', '姓名是' + name + ',今年' + age + '岁了')
```

适合简易场景



总结

1. 模板字符串的作用？优点是什么？

拼接 **字符串** 和 **变量**，更适合**多个变量**的拼接场景

2. 使用的符号是什么？

``` 需要变量用 `${ ... }` 包裹

```
let name: string = '小明'
let age: number = 18
console.log('简介信息', `姓名是${name}, 今年${age}岁了`)
```

# 类型转换（数字和字符串）



黑马程序员  
[www.itheima.com](http://www.itheima.com)

传智教育旗下  
高端IT教育品牌

## 类型转换（数字和字符串）



## 类型转换（数字和字符串）

### 1. 字符串转数字

**Number():** 字符串 直接转数字，转换失败返回NaN（字符串中包含非数字）

**parseInt():** 去掉小数部分 转数字，转换失败返回NaN

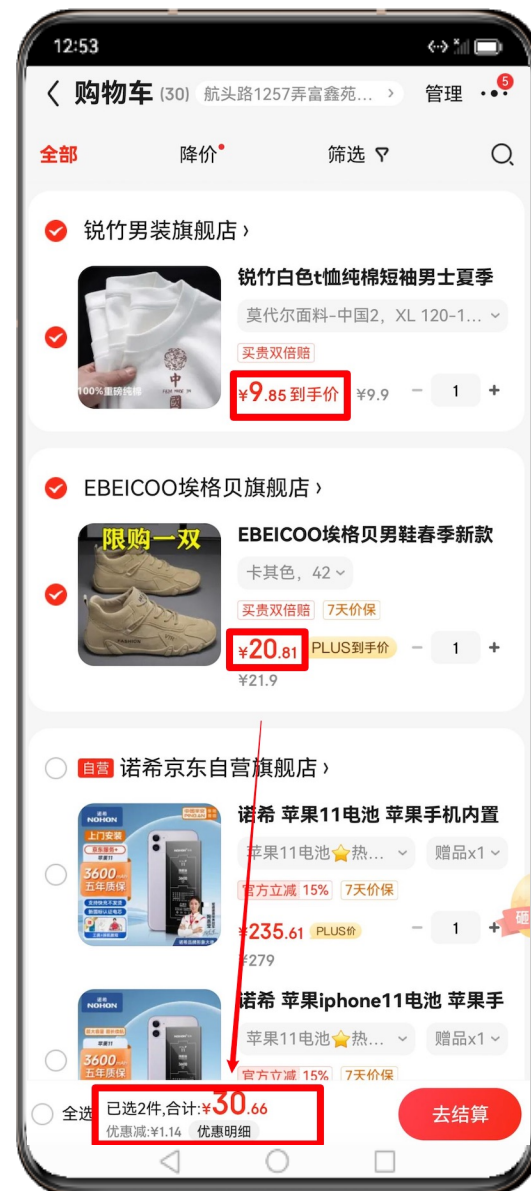
**parseFloat():** 保留小数部分 转数字，转换失败返回NaN

▼ Number

ArkTS

复制代码

```
1 let str1: string = '1.1'
2 let str2: string = '1.9'
3 let str3: string = '1.1a'
4
5 console.log('数字是', Number(str1)) // 1.1
6 console.log('数字是', Number(str2)) // 1.9
7 console.log('数字是', Number(str3)) // NaN
```



## 类型转换（数字和字符串）

### 1. 字符串转数字

**Number():** 字符串 直接转数字，转换失败返回NaN（字符串中包含非数字）

**parseInt():** 去掉小数部分 转数字，转换失败返回NaN

**parseFloat():** 保留小数部分 转数字，转换失败返回NaN

▼ parseInt

ArkTS

复制代码

```
1 let str1: string = '1.1'
2 let str2: string = '1.9'
3 let str3: string = '1.1a'
4 let str4: string = 'a'
5
6 console.log('数字是', parseInt(str1)) // 1
7 console.log('数字是', parseInt(str2)) // 1
8 console.log('数字是', parseInt(str3)) // 1
9 console.log('数字是', parseInt(str4)) // NaN
```



## 类型转换（数字和字符串）

### 1. 字符串转数字

**Number()**: 字符串 直接转数字，转换失败返回NaN（字符串中包含非数字）

**parseInt()**: 去掉小数部分 转数字，转换失败返回NaN

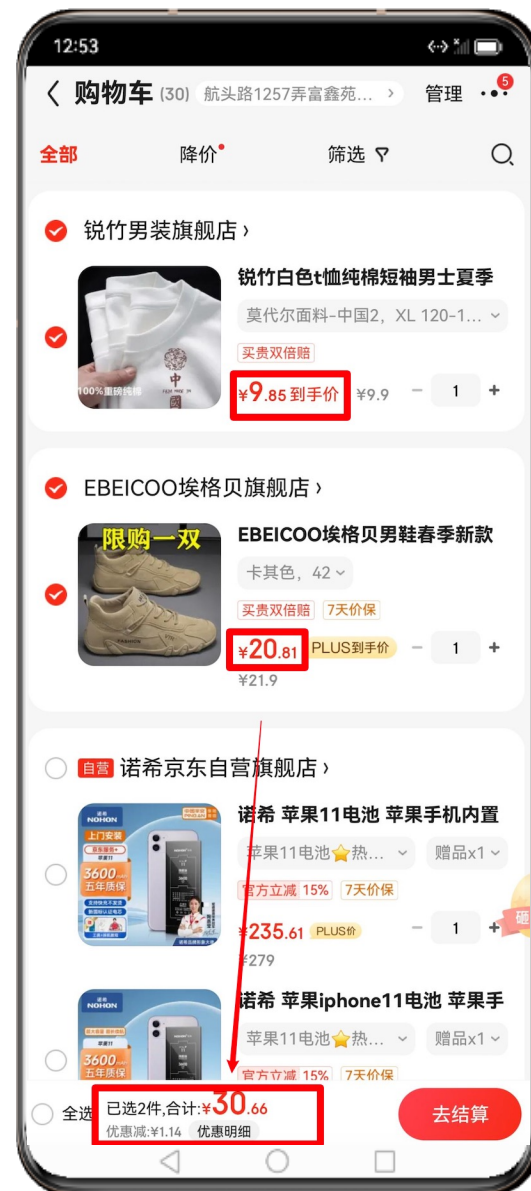
**parseFloat()**: 保留小数部分 转数字，转换失败返回NaN

▼ parseFloat

ArkTS

复制代码

```
1 let str1: string = '1.1'
2 let str2: string = '1.9'
3 let str3: string = '1.1a'
4 let str4: string = 'a'
5
6 console.log('数字是', parseFloat(str1)) // 1.1
7 console.log('数字是', parseFloat(str2)) // 1.9
8 console.log('数字是', parseFloat(str3)) // 1.1
9 console.log('数字是', parseInt(str4)) // NaN
```



## 类型转换（数字和字符串）

### 2. 数字转字符串

**toString():** 数字直接转字符串

**toFixed():** 四舍五入转字符串，可设置保留几位小数

#### toString

ArkTS | 复制代码

```
1 let num1: number = 1.1
2 let num2: number = 1.9
3
4 console.log('字符串是', num1.toString()) // '1.1'
5 console.log('字符串是', num2.toString()) // '1.9'
```

#### toFixed

ArkTS | 复制代码

```
1 let num1: number = 1.1
2 let num2: number = 1.9
3 let num3: number = 1.9152
4
5 console.log('字符串是', num1.toFixed()) // '1'
6 console.log('字符串是', num2.toFixed()) // '2'
7 console.log('字符串是', num3.toFixed(2)) // '1.92'
```

```
let num1: number = 100

@Entry
@Component
struct Index {
 build() {
 Column() {
 Text(num1)
 }
 }
}
```

Text 需要 字符串类型 渲染



# 总结

## 1. 字符串转数字?

**Number()**: 字符串 直接转数字，转换失败返回NaN (**字符串中包含非数字**)

**parseInt()**: 去掉小数部分 转数字，转换失败返回NaN

**parseFloat()**: 保留小数部分 转数字，转换失败返回NaN

## 2. 数字转字符串?

**toString()**: 数字直接转字符串

**toFixed()**: 四舍五入转字符串，可设置保留几位小数



# 交互 - 点击事件



黑马程序员  
[www.itheima.com](http://www.itheima.com)

传智教育旗下  
高端IT教育品牌

## 交互 - 点击事件

说明：组件 **被点击时** 触发的事件

作用：监听（感知）用户 **点击行为**，进行对应**操作**

语法：onClick( (参数) => {} )

```
Button('点我, 显示弹框')
 .onClick(() => {
 AlertDialog.show({
 message: '你好-这是个弹框'
 })
 })
})
```





# 总结

1. 如何监听用户点击事件?

语法: `onClick( (参数) => {} )`

2. 如何在界面中弹出一个对话框?

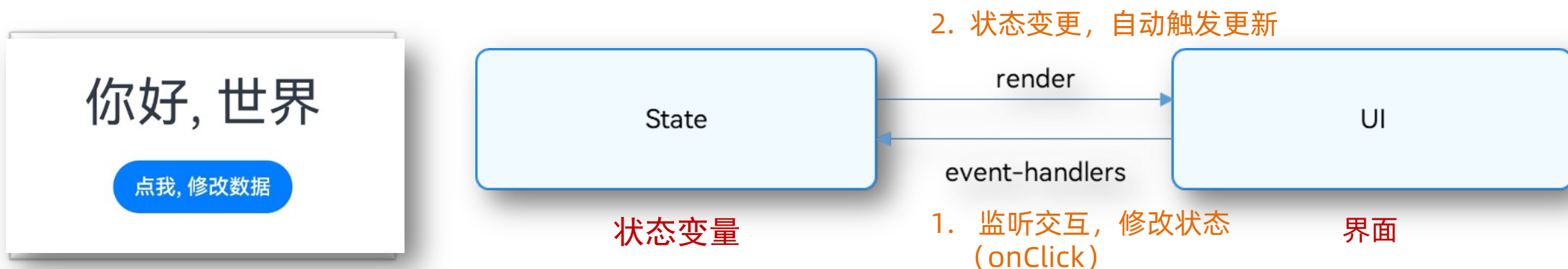
```
Button('点我, 显示弹框')
 .onClick(() => {
 AlertDialog.show({
 message: '你好-这是个弹框'
 })
 })
```

# 状态管理

## 状态管理

之前构建的页面多为静态界面。

但如果希望构建一个动态的、有交互的界面，就需要引入“状态”的概念



点击交互 触发了 文本状态变更，状态变更 引起了 UI渲染

## 状态管理

普通变量：只能在**初始化**时渲染，后续将不会再刷新。

**状态变量**：需要**装饰器**装饰，**改变**会引起 UI 的渲染刷新（必须设置 **类型** 和 **初始值**）

```
let msg1: string = '黑马程序员'

@Entry
@Component
struct Index {
 msg2: string = '学鸿蒙,来黑马'
 build() {
 Column() {
 Text(msg1)
 Text(this.msg2)
 }
 }
}
```

普通变量

```
@Entry
@Component
struct Index {
 @State msg3: string = 'Hello World'
 build() {
 Column() {
 Text(this.msg3).onClick(() => {
 this.msg3 = '你好, 世界'
 })
 }
 }
}
```

状态变量

注意：定义在 **组件内** 普通变量 或 状态变量，都需要 **通过 this 访问**

# 总结

## 1. 状态管理的基本流程图



## 2. 普通变量 和 状态变量 的区别？

普通变量：只能在初始化时渲染，后续将不会再刷新。

状态变量：需要装饰器装饰，改变会引起 UI 的渲染刷新。

```
let msg1: string = '黑马程序员'

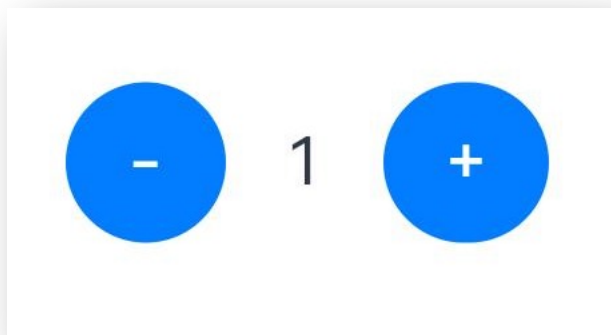
@Entry
@Component
struct Index {
 msg2: string = '学鸿蒙,来黑马'
 build() {
 Column() {
 Text(msg1)
 Text(this.msg2)
 }
 }
}
```

普通变量

```
@Entry
@Component
struct Index {
 @State msg3: string = 'Hello World'
 build() {
 Column() {
 Text(this.msg3).onClick(() => {
 this.msg3 = '你好,世界'
 })
 }
 }
}
```

状态变量

## 案例 计数器案例



核心思路：

1. 准备 状态变量 → `@State count: number = 1`
2. 注册 点击事件 → `onClick`
3. 点击时，修改 状态变量
4. 状态变量变化，界面自动更新



# 运算符



黑马程序员  
[www.itheima.com](http://www.itheima.com)

传智教育旗下  
高端IT教育品牌


## 算数运算符

算术运算符：也叫数学运算符，主要包括加、减、乘、除、取余（求模）等

算数运算符	作用
+	加法运算
-	减法运算
*	乘法运算
/	除法运算
%	取余（求模）

### 算数运算符

ArkTS

 复制代码

```
1 let num1: number = 9
2 let num2: number = 4
3
4 console.log('加法计算', num1 + num2)
5 console.log('减法计算', num1 - num2)
6 console.log('乘法计算', num1 * num2)
7 console.log('除法计算', num1 / num2) // 2.25
8 console.log('取余计算', num1 % num2) // 1
```


## 赋值运算符

赋值运算符：对变量进行 **赋值** 的运算符，如：=

赋值运算符	作用
<b>+=</b>	<b>加法赋值</b>
<b>-=</b>	减法赋值
<b>*=</b>	乘法赋值
<b>/=</b>	除法赋值
<b>%=</b>	取余赋值

### ▼ 赋值运算符

ArkTS

 复制代码

```
1 let num1: number = 1
2
3 num1 += 1
4
5 console.log('加等后的结果', num1)
```

## 总结

### 算数运算符 和 赋值运算符?

算数运算符	作用
+	加法运算
-	减法运算
*	乘法运算
/	除法运算
%	取余 (求模)

赋值运算符	作用
+=	加法赋值
-=	减法赋值
*=	乘法赋值
/=	除法赋值
%=	取余赋值

#### 算数运算符

ArkTS

```
1 let num1: number = 9
2 let num2: number = 4
3
4 console.log('加法计算', num1 + num2)
5 console.log('减法计算', num1 - num2)
6 console.log('乘法计算', num1 * num2)
7 console.log('除法计算', num1 / num2) // 2.25
8 console.log('取余计算', num1 % num2) // 1
```

#### 赋值运算符

ArkTS

复制代码

```
1 let num1: number = 1
2
3 num1 += 1
4
5 console.log('加等后的结果', num1)
```

## 案例 点赞案例



考眼力又来了你能看到  
几只鸭子?

视野联行眼镜 8888

核心思路分析:

1. 注册点击事件 → **onClick**
2. 点击时候, 修改 **颜色**, 修改 **数字**
  - ① 提取 **颜色** 为状态变量
  - ② 提取 **数字** 为状态变量

## 一元运算符

常见一元运算符：++ 和 --

- 后置写法：先赋值后自增/自减
- 前置写法：先自增/自减再赋值

```
let num: number = 10
let res: number = num++ // 后自增
```

```
let num2: number = 10
let res2: number = ++num // 先自增
```



# 总结

常见一元运算符：++ 和 --，前置后置写法区别？

1. 后置写法：先赋值后自增/自减

2. 前置写法：先自增/自减再赋值

```
let num: number = 10
let res: number = num++ // 后自增
```

```
let num2: number = 10
let res2: number = ++num // 先自增
```

## 比较运算符

作用：用来 **判断比较** 两个数据 **大小**，返回一个布尔值 (**true / false**)

比较运算符	作用
>	判断大于
>=	判断大于等于
<	判断小于
<=	判断小于等于
==	判断相等
!=	判断不相等

姓名	语文	数学	英语
小华	32	95	55
小丽	92	61	85

```
let num1: number = 9
let num2: number = 5

console.log('比较运算的结果是', num1 > num2)
console.log('比较运算的结果是', num1 >= num2)
console.log('比较运算的结果是', num1 == num2)
console.log('比较运算的结果是', num1 != num2)
```





# 总结

比较运算符	作用
>	判断大于
>=	判断大于等于
<	判断小于
<=	判断小于等于
==	判断相等
!=	判断不相等

```
let num1: number = 9
let num2: number = 5

console.log('比较运算的结果是', num1 > num2)
console.log('比较运算的结果是', num1 >= num2)
console.log('比较运算的结果是', num1 == num2)
console.log('比较运算的结果是', num1 != num2)
```

## 逻辑运算符

作用：扩充判断条件

逻辑运算符	作用
&&	与，都真才真
	或，一真则真
!	非，取反

```
let num1: number = 9
let num2: number = 5
let num3: number = 3

console.log('结果是', num1 > num2 && num2 > num3)
console.log('结果是', num1 > num2 && num2 < num3)
console.log('结果是', num1 > num2 || num2 < num3)
console.log('结果是', !true)
```

账号登录

短信登录

×

手机号/用户名/邮箱

密码

忘记密码?

登录

☐ 阅读并接受 [百度用户协议](#) 和 [隐私政策](#)



# 总结

## 逻辑运算符?

逻辑运算符	作用
&&	与，都真才真
	或，一真则真
!	非，取反

```
let num1: number = 9
let num2: number = 5
let num3: number = 3

console.log('结果是', num1 > num2 && num2 > num3)
console.log('结果是', num1 > num2 && num2 < num3)
console.log('结果是', num1 > num2 || num2 < num3)
console.log('结果是', !true)
```

## 运算符优先级

优先级	顺序
1 小括号	()
2 一元	++、--、!
3 算数	先 *、/、% 后 +、-
4 比较	>、>=、<、<=
5 比较	==、!=
6 逻辑运算符	先 && 后
7 赋值	=

规则：

1. 小括号
2. 一元
3. 算数
4. 比较
5. 逻辑
6. 赋值

# 综合案例 – 美团购物车



黑马程序员  
[www.itheima.com](http://www.itheima.com)

传智教育旗下  
高端IT教育品牌

## 美团购物车

需求分析：

1. 商品区域：数字框 + -
2. 底部结算：联动计算 并 渲染展示
  - ① 已选件数
  - ② 总价格
  - ③ 优惠价格

核心思路：

1. 提取状态：数量、原价、现价
2. 界面绑定
3. 点击修改数据，自动更新





传智教育旗下高端IT教育品牌