JavaScript 核心第六天

箭头函数&解构&常用方法

黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌

学习目标

Learning Objectives

学习ES6新特性让代码书写更加简洁便利

目录

Contents

函数进阶

- 箭头函数
- 函数提升
- 函数参数

箭头函数（重要）

箭头函数比函数表达式更简洁的一种写法

使用场景：箭头函数更适用于那些本来需要匿名函数的地方，写法更简单

基本写法

```
// 普通函数
const fn = function () {
  console.log('我是普通函数')
}
fn()
```

```
// 箭头函数
const fn = () => {
  console.log('俺是箭头函数')
}
fn()
```

箭头函数（重要）

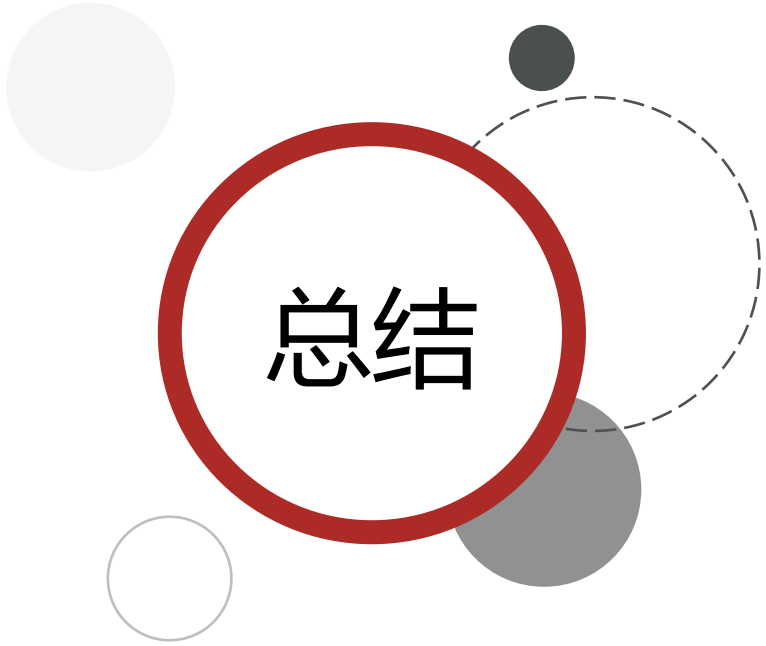
用法细节：

1. 当箭头函数只有一个参数时，可以省略参数的小括号，其余个数不能省略（没有参数也需要写小括号）
2. 当箭头函数的函数体只有一句代码 可以省略函数体大括号，这句代码就是返回值（可以不用写return）
3. 如果返回的是个对象，则需要把对象用小括号包裹

```
const sum = x => {
  console.log(x + x)
}
```

```
const sum = x => x + x
console.log(sum(5))
```

```
const fn = () => ({ age: 18 })
console.log(fn())
```



1. 箭头函数的语法

   () => {}

2. 箭头函数的使用细节

   - 有且仅有一个形参，可以省略小括号不写
   - 函数体只有一句代码，可以省略 { } 和 return
   - 函数体返回对象，需要使用小括号包裹对象

函数进阶

- 函数提升
- 函数参数
- 箭头函数

函数参数

函数参数的使用细节，能够提升函数应用的灵活度。

学习路径：

1. arguments对象（了解）
2. 剩余参数（重点）

函数参数

1. arguments对象（了解）

箭头函数里面没有 arguments

arguments 是函数内部内置的对象，它包含了调用函数时传入的所有实参

使用场景：

写一个求和函数，不管多少实参都可以求结果。 问题是形参怎么写？

```
sum(1, 2)
sum(1, 2, 3)
sum(1, 2, 3, 4)
```

```
function sum(形参怎么写) {
  // 业务代码
}
```

函数参数

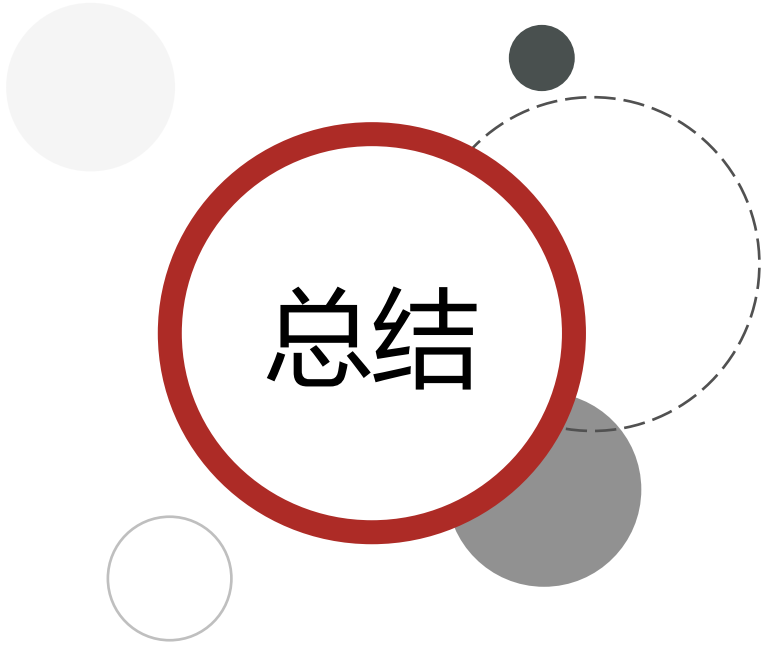
arguments对象（了解）

arguments 是函数内部内置的对象，它包含了调用函数时传入的所有实参

```
// arguments对象获取所有实参
function sum() {
  // 1. arguments 只存在于函数中 伪数组
  // 2. arguments 可以得到传递过来的所有实参 [1, 2]
  // console.log(arguments)
  let sum = 0
  for (let i = 0; i < arguments.length; i++) {
    sum += arguments[i]
  }
  console.log(sum)
}
sum(1, 2)
sum(1, 2, 3)
sum(1, 2, 3, 4)
// console.log(arguments) 外面无法使用
```

总结：

1. arguments 只存在于函数中（但不能在箭头函数中使用）
2. arguments 的作用是动态获取函数的实参
3. 可以通过for循环依次得到传递过来的实参



1. 当不确定传递多少个实参的时候，我们怎么办？
   - arguments 对象，它可以得到传递过来的所有实参
   - 它只存在函数中
2. arguments 对象不能在哪里使用？
   - 函数外
   - 箭头函数

补充 - 伪数组

arguments 是一个**伪数组**：

- 有长度有索引号的数组
- 但是没有 pop() push() 等数组方法

```
function fn() {
  console.log(arguments)

  arguments.push(60)
}
fn(10, 20, 30)
```

```
Uncaught TypeError: arguments.push is not a function
    at fn (classList.html:29:17)
    at classList.html:31:5
```

函数参数

函数参数的使用细节，能够提升函数应用的灵活度。

学习路径：

1. arguments对象（了解）
2. 剩余参数（重点）

函数参数

剩余参数（重点）

剩余参数:允许我们将一个不定数量的参数表示为一个数组

简单理解：用于获取多余的实参，并形成一个真数组

使用场景：

也可以解决形参和实参个数不匹配的问题

```
function sum(...other) {
  // 可以得到 [1, 2, 3, 4]
  console.log(other)
}
sum(1, 2, 3, 4)
```

那和arguments 有什么不同吗?

函数参数

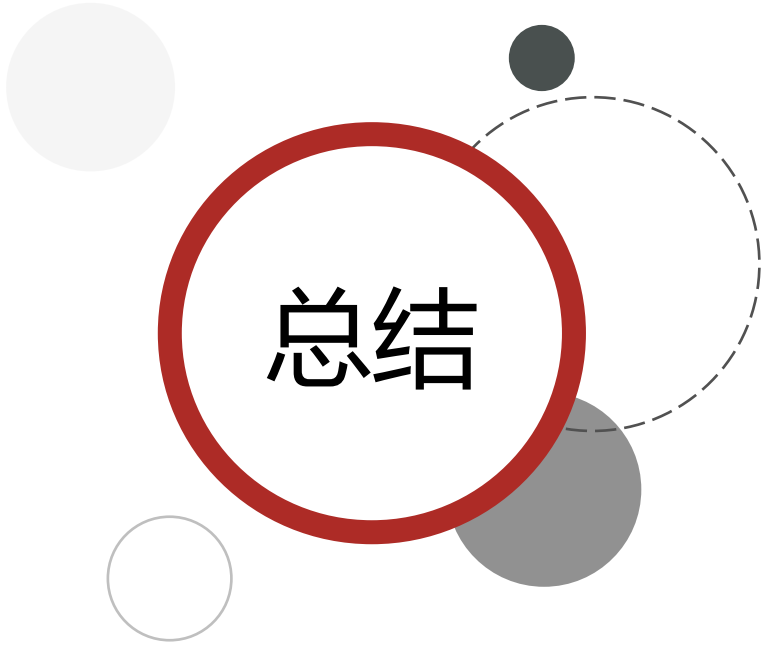
剩余参数和arguments区别

1. ... 是语法符号，置于最末函数形参之前，用于获取多余的实参

2. 借助 ... 获取的剩余实参，是个真数组

3. 箭头函数不支持arguments，但是可以使用剩余参数

开发中，还是提倡多使用 **剩余参数**



1. 剩余参数主要的使用场景是?
   - 用于获取多余的实参，返回一个真数组
2. 剩余参数和arguments区别是什么？开发中提倡使用哪一个?
   - arguments 是伪数组
   - 剩余参数是真数组
   - 箭头函数不支持arguments，但是可以使用剩余参数
   - 开发中使用剩余参数想必也是极好的

展开运算符

展开运算符(...),将一个数组/对象进行展开

语法:

1. 不会修改原数组

```
const arr = [1, 5, 3, 8, 2]
console.log(...arr) // 1 5 3 8 2
```

典型运用场景： 求数组最大值(最小值)、合并数组等

```
const arr = [1, 5, 3, 8, 2]
// console.log(...arr) // 1 5 3 8 2
console.log(Math.max(...arr)) // 8
console.log(Math.min(...arr)) // 1
```

```
// 合并数组
const arr1 = [1, 2, 3]
const arr2 = [4, 5, 6]
const arr3 = [...arr1, ...arr2]
console.log(arr3) // [1,2,3,4,5,6]
```

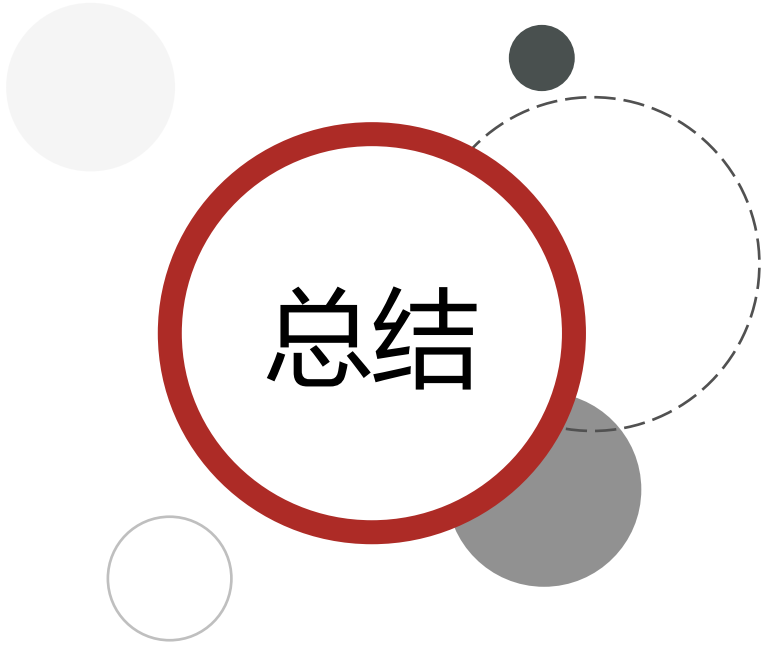
展开运算符 or 剩余参数

剩余参数：函数参数使用，把多个元素收集起来生成一个真数组 （凝聚）

展开运算符：将数组展开成各个元素（拆散）

```
function getSum(...other) {
  // other 得到 [1,2,3]
  console.log(other)
}
getSum(1, 2, 3)
```

```
const arr = [1, 5, 3, 8, 2]
console.log(...arr) // 1 5 3 8 2
```



1. 展开运算符主要的作用是?
   - 可以把数组展开,可以利用求数组最大值以及合并数组等操作
2. 展开运算符和剩余参数有什么区别?
   - 展开运算符主要是 数组展开（拆散）
   - 剩余参数 在函数内部使用，把多个元素收集起来生成一个真数组（凝聚）

```
const arr = [1, 5, 3, 8, 2]
console.log(...arr) // 1 5 3 8 2
```

```
function getSum(...other) {
  // other 得到 [1,2,3]
  console.log(other)
}
getSum(1, 2, 3)
```

目录

Contents

解构赋值

- 数组解构
- 对象解构

解构赋值

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

解构：其实就是把一个事物的结构进行拆解

使用场景：

```
const arr = [100, 60, 80]
console.log(arr[0]) // 最大值
console.log(arr[1]) // 最小值
console.log(arr[2]) // 平均值
```

```
const arr = [100, 60, 80]
const max = arr[0]
const min = arr[1]
const avg = arr[2]
console.log(max) //最大值 100
console.log(min) // 最小值 60
console.log(avg) // 平均值 80
```

```
const [max, min, avg] = [100, 60, 80]
console.log(max) //最大值 100
console.log(min) // 最小值 60
console.log(avg) // 最小值 80
```

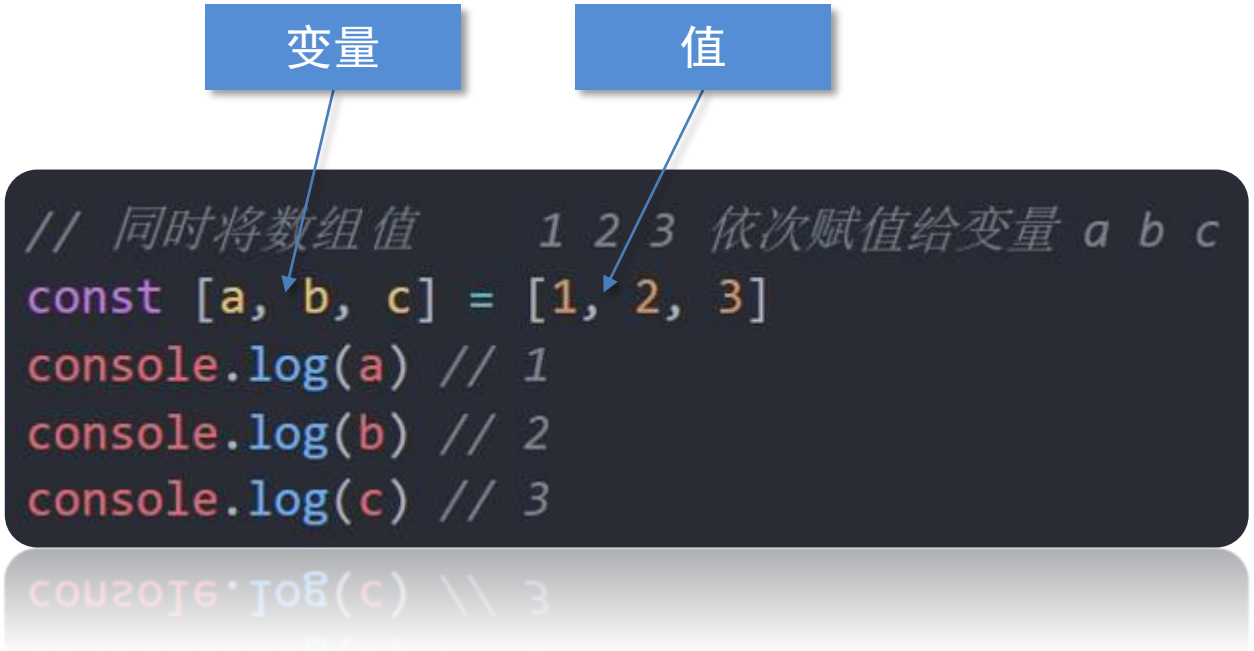
解构写法

数组解构

基本语法：

1. 右侧数组的值将被赋值给左侧的变量

2. 变量的顺序对应数组值的位置依次进行赋值操作



数组解构

典型应用：交换2个变量

```
let a = 1
let b = 3;
[b, a] = [a, b]
console.log(a) // 3
console.log(b) // 1
```

这里必须有分号

数组解构

注意： js 前面必须加分号情况

1. 小括号开头

```
(function t() { })();
// 或者
;(function t() { })()
```

2. 中括号开头

```
// 数组开头的，特别是前面有语句的一定注意加分号
;[b, a] = [a, b]
```

```
let a = 1
let b = 3;
[b, a] = [a, b]
console.log(a) // 3
console.log(b) // 1
```

```
const [max, min, avg] = [100, 60, 80]
console.log(max) //最大值 100
console.log(min) // 最小值 60
console.log(avg) // 最小值 80
```

总结

1. 解构赋值有什么作用?
   - 可以将数组中的值或对象的属性取出，赋值给其他变量
2. Js 前面有两哪种情况需要加分号的?
   - 小括号开头
   - 中括号开头

```
(function t() { })();
// 或者
;(function t() { })()
```

```
// 数组开头的，特别是前面有语句的一定注意加分号
;[b, a] = [a, b]
```

练习

独立完成数组解构赋值

需求①： 有个数组： const pc = ['海尔', '联想', '小米', '方正']

解构为变量: hr lx mi fz

需求②：请将最大值和最小值函数返回值解构 max 和min 两个变量

```
function getValue() {
  return [100, 60]
}

// 要求 max 变量里面存100  min 变量里面存 60
```

数组解构

数组解构：变量和值不匹配的情况

1. 变量多值少的情况：

```
// 变量多，值少
const [a, b, c, d] = ['小米', '苹果', '华为']
console.log(a) // 小米
console.log(b) // 苹果
console.log(c) // 华为
console.log(d) // undefined
```

变量的数量大于值数量时，多余的变量将被赋值为 undefined

数组解构

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

2. 防止有undefined传递值的情况，可以设置默认值：

```
const [a = '手机', b = '华为'] = ['小米']
console.log(a) // 小米
console.log(b) // 华为
```

数组解构

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

3. 变量少值多的情况：

```
// 变量少，值多
const [a, b, c] = ['小米', '苹果', '华为', '格力']
console.log(a) // 小米
console.log(b) // 苹果
console.log(c) // 华为
```

数组解构

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

4. 利用剩余参数解决变量少值多的情况：

```
// 利用剩余参数 变量少,值多
const [a, b, ...tel] = ['小米', '苹果', '华为', '格力', 'vivo']
console.log(a) // 小米
console.log(b) // 苹果
console.log(tel) // ['华为', '格力', 'vivo']
```

数组解构

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

5. 按需导入，忽略某些值：

```
// 按需导入，忽略某些值
const [a, , c, d] = ['小米', '苹果', '华为', '格力']
console.log(a) // 小米
console.log(c) // 华为
console.log(d) // 格力
```

数组解构

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

6. 支持多维数组的解构：

```
const [a, b] = ['苹果', ['小米', '华为']]
console.log(a) // 苹果
console.log(b) // ['小米', '华为']
```

```
// 想要拿到 小米和华为怎么办？
const [a, [b, c]] = ['苹果', ['小米', '华为']]
console.log(a) // 苹果
console.log(b) // 小米
console.log(c) // 华为
```

解构赋值

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

分为：

- 数组解构
- 对象解构

对象解构

对象解构赋值：可以将对象的属性取出，赋值给其他变量

使用场景：

以前使用对象属性提取比较麻烦

有了对象解构赋值语法就简洁了很多

```
const user = {
  username: '小明',
  age: 18,
  gender: '男'
}
```

```
// 使用属性
user.username
user.age
user.gender
```

```
// 对象解构赋值
const user = {
  username: '小明',
  age: 18,
  gender: '男'
}

const { username, age, gender } = user
// 使用属性
username   // 小明
age  // 18
gender  // 男
```

对象解构

1. **基本语法：**

右侧对象的属性值将被赋值给左侧的变量

注意：

- 对象的属性名一定要和变量名相同
- 变量名如果没有和对象属性名相同的则默认是 undefined
- 注意解构的变量名不要和外面的变量名冲突否则报错

```
// 对象解构赋值
const user = {
  username: '小明',
  age: 18,
  gender: '男'
}
const { username, age, gender } = user
// 使用属性
console.log(username)  // 小明
console.log(age)  // 18
console.log(gender)  // 男
```

练习

独立完成对象数组解构赋值

需求①： 有个对象： const pig = { name: '佩奇',age: 6 }

结构为变量：完成对象解构，并以此打印出值

对象解构

2.更改解构变量名：

可以从一个对象中提取变量并同时修改新的变量名

```
// 普通对象
const user = {
  name: '小明',
  age: 18
};
// 把 原来的name 变量重新命名为 uname
const { name: uname, age } = user
console.log(uname) // 小明
console.log(age) // 18
```

变量名：新变量名

对象解构

3. 对象数组解构

```
const pig = [
  {
    name: '佩奇',
    age: 6
  }
]
```

```
const [{ name, age }] = pig
console.log(name, age)
```

练习

独立完成对象数组解构赋值

需求①： 有个对象： const pig = { name: '佩奇',age: 6 }

结构为变量：完成对象解构，并以此打印出值

需求②：请将pig对象中的name，通过对象解构的形式改为 uname，并打印输出

需求③：请将 数组对象，完成 商品名和价格的解构

```
const goods = [
      {
        goodsName: '小米',
        price: 1999
      }
]
```

对象解构

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

3. 多级对象解构：

```
const pig = {
  name: '佩奇',
  family: {
    mother: '猪妈妈',
    father: '猪爸爸',
    sister: '乔治'
  },
  age: 6
}
```

```
// 依次打印家庭成员
const pig = {
  name: '佩奇',
  family: {
    mother: '猪妈妈',
    father: '猪爸爸',
    sister: '乔治'
  },
  age: 6
}
const { name, family: { mother, father, sister } } = pig
console.log(name) // 佩奇
console.log(mother) // 猪妈妈
console.log(father) // 猪爸爸
console.log(sister) // 乔治
```

对象解构

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

3. 多级对象解构：

```
const person = [
  {
    name: '佩奇',
    family: {
      mother: '猪妈妈',
      father: '猪爸爸',
      sister: '乔治'
    },
    age: 6
  }
]
```

```
const [{ name, family: { mother, father, sister } }] = people
console.log(name) // 佩奇
console.log(mother) // 猪妈妈
console.log(father) // 猪爸爸
console.log(sister) // 乔治
```

对象解构

解构赋值：可以将数组中的值或对象的属性取出，赋值给其他变量

3. 多级对象解构：

```
// 这是后台传递过来的数据，请选出data里面的数据方便后面渲染W
const msg = {
  "code": 200,
  "msg": "获取新闻列表成功",
  "data": [
    {
      "id": 1,
      "title": "5G商用自己，三大运用商收入下降",
      "count": 58
    },
    {
      "id": 2,
      "title": "国际媒体头条速览",
      "count": 56
    },
    {
      "id": 3,
      "title": "乌克兰和俄罗斯持续冲突",
      "count": 1669
    },

  ]
}
```

```
const { data } = msg
console.log(data)
```

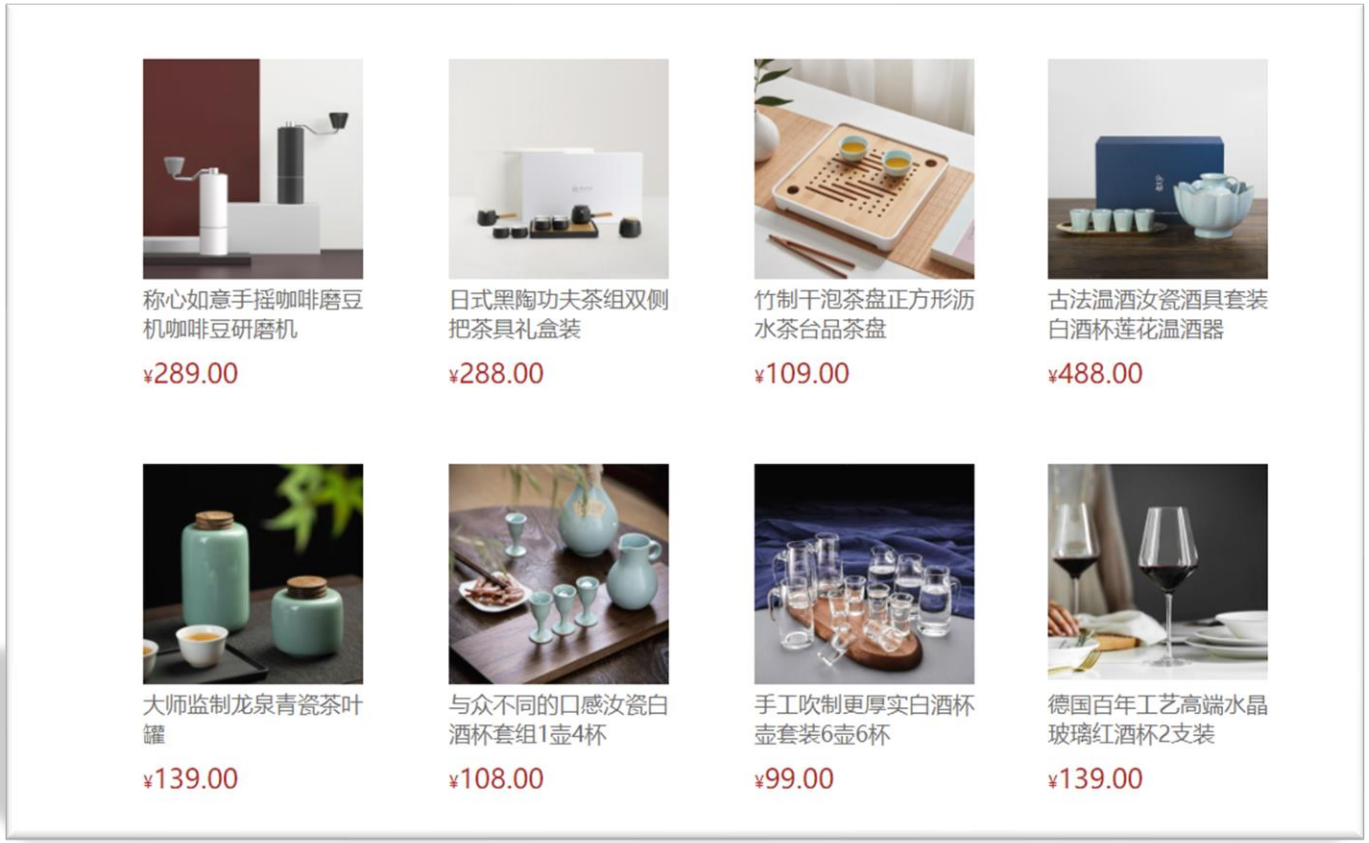
```
// 2. 需求， 请将上面对象只选出 data数据，传递给另外一个函数
function render({ data }) {
  // 内部处理
  console.log(data)
}
render(msg)
```

```
// 3. 需求， 为了防止msg里面的data名字混淆，要求渲染函数里面的数据名改为 myData
function render({ data: myData }) {
  // 内部处理
  console.log(myData)
}
render(msg)
```

案例

渲染商品列表案例

请根据数据渲染以下效果



步骤

渲染商品列表案例

核心思路：有多少条数据，就渲染多少模块，然后 生成对应的 html结构标签， 赋值给 list标签即可

①：拿到数据，利用for循环 和 **拼接字符串**生成结构添加到页面中

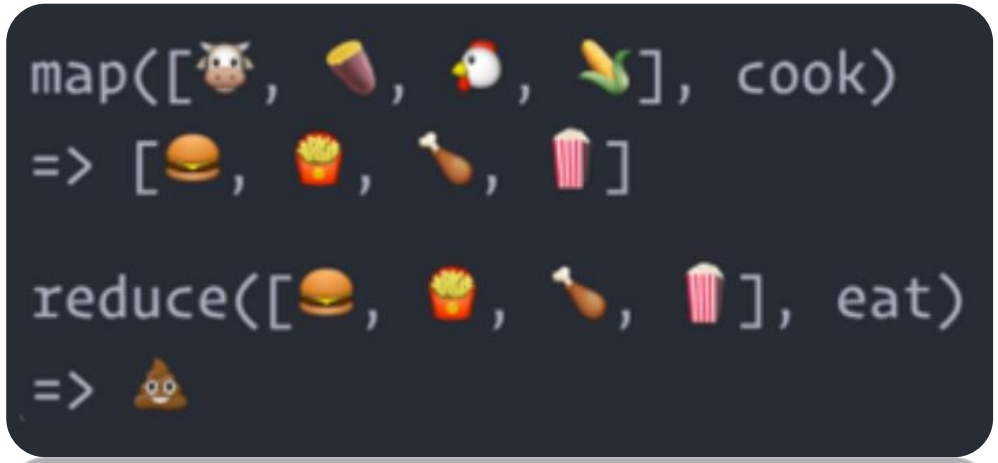
②：填充数据的时候使用 解构赋值

目录

Contents

数组常见方法

方法	作用	说明
forEach	遍历数组	不返回数组，经常用于查找遍历数组元素
map	迭代数组	返回新数组，返回的是处理之后的数组元素，想要使用返回的新数组
join	拼接数组元素	返回指定连接符拼接得到的字符串
reduce	累计器	返回累计处理的结果，经常用于求和等



数组的forEach方法

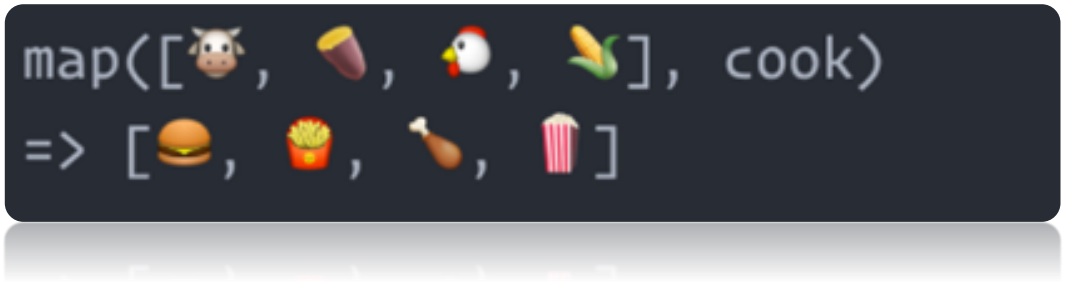
- **使用场景：**

forEach 可以遍历数组

```
数组.forEach(function (element, index) {
  // element 是数组元素
  // index 是数组元素的索引号
})
```

可以配合箭头函数，使代码更简洁

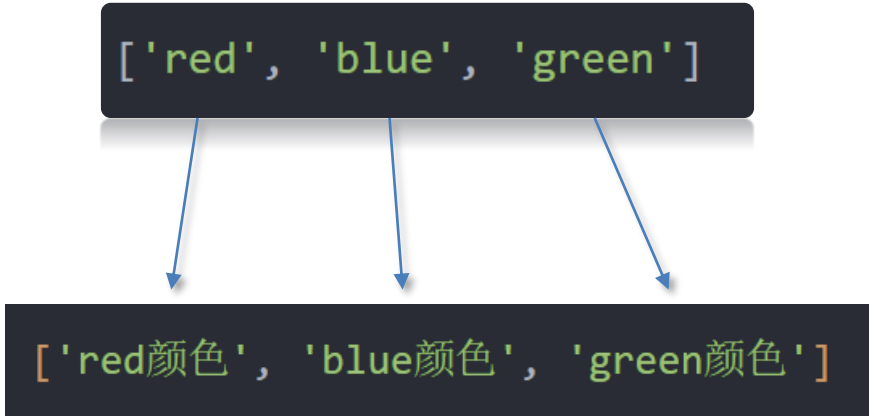
数组的map方法



- **使用场景：**

map 可以遍历数组处理数据，并且**返回新的数组**

```
const arr = ['red', 'blue', 'green']
const newArr = arr.map(function (ele, index) {
  console.log(ele)  // 数组元素
  console.log(index) // 数组索引号
  return ele + '颜色'
})
console.log(newArr)   //  ['red颜色', 'blue颜色', 'green颜色']
```



map 也称为映射。映射是个术语，指两个元素的集之间元素相互“对应”的关系。

map重点在于有返回值，forEach没有返回值（undefined）

数组的join方法

- **作用：**

  join() 方法用于把数组中的所有元素连接成一个字符串

- **语法：**

```
const arr = ['red颜色', 'blue颜色', 'green颜色']
console.log(arr.join(''))  // red颜色blue颜色green颜色
```

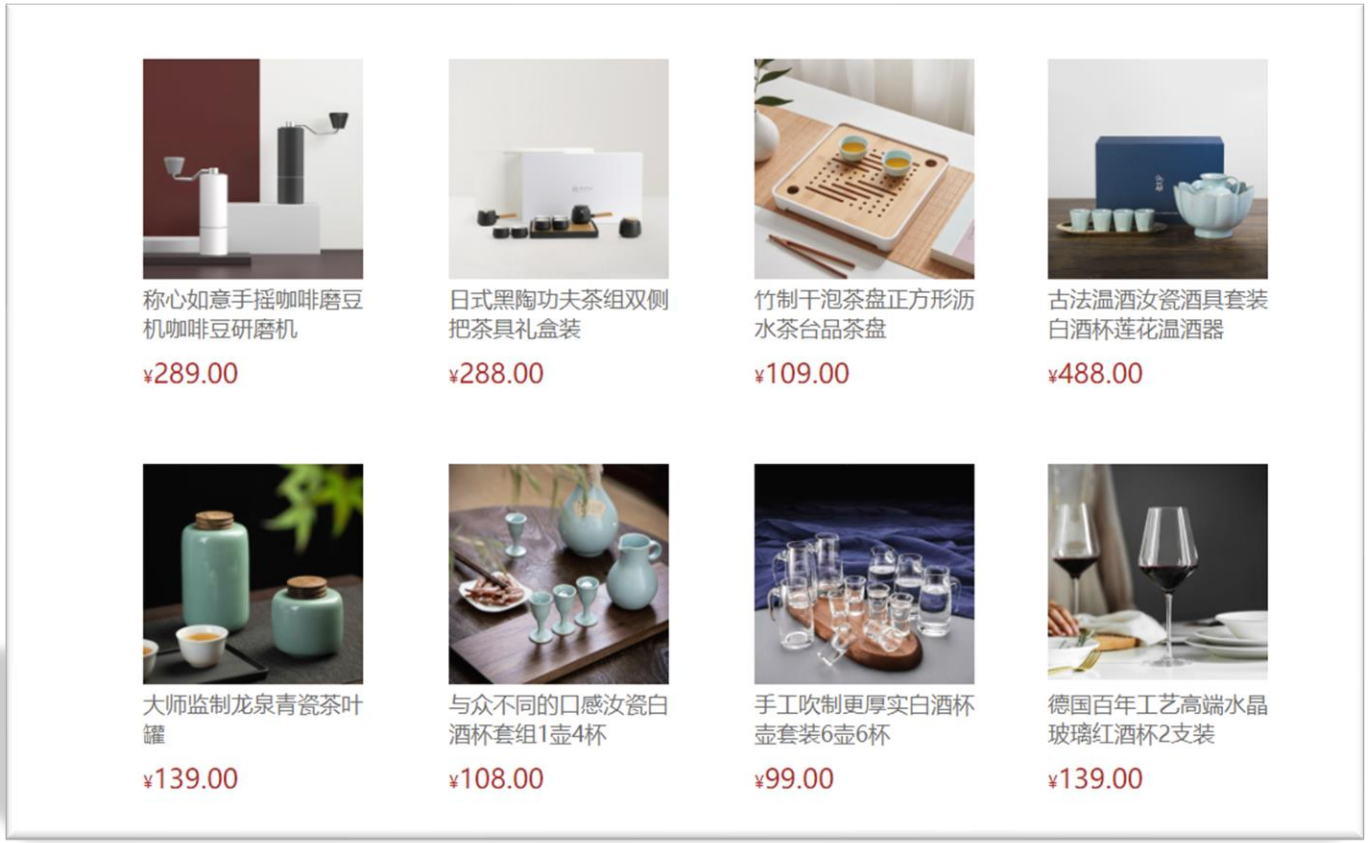
- **参数：**

  数组元素是通过参数里面指定的分隔符进行分隔的，空字符串（''），则所有元素之间都没有任何字符。

案例

渲染商品列表案例

使用 map + join 方法 渲染商品列表

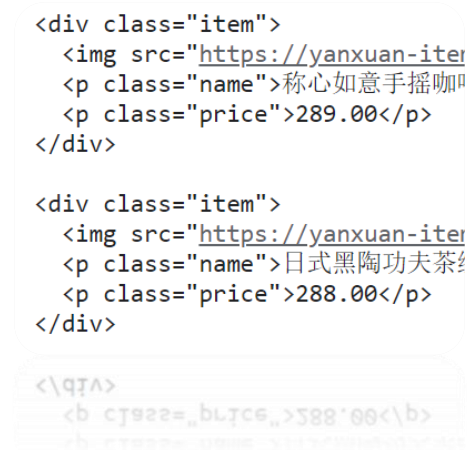


数组的map + join 方法渲染页面思路：

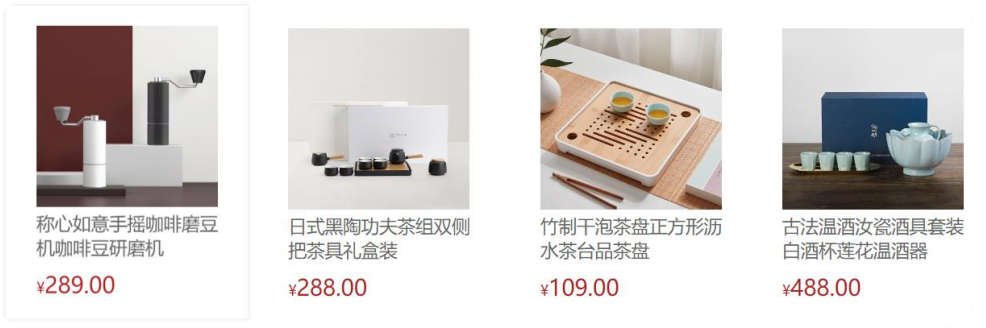
map遍历数组处理数据生成 . Item,
返回一个数组

把数组转换为字符串

设置给 .list




```
const goodsList = [
  {
    id: '4001172',
    name: '称心如意手摇咖啡磨豆机咖啡豆研磨机',
    price: '289.00',
    picture: 'https://yanxuan-item.nosdn.1
  }
]
```



数组的 reduce 方法

- **作用：**reduce 返回累计处理的结果，经常用于求和等
- **基本语法：**

```
arr.reduce(function(){}, 起始值)
```

```
arr.reduce(function(上一次值, 当前值){}, 初始值)
```

```
const arr = [1, 2, 3, 4]
```

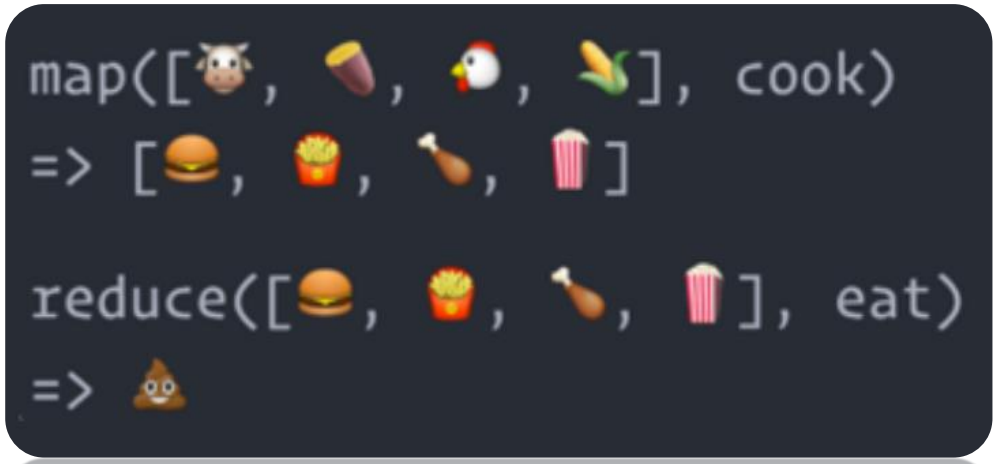
- **参数：**

1. 如果有起始值，则把初始值累加到里面

数组的 reduce 方法

- **reduce 执行过程：**
  1. 如果没有起始值， 则上一次值以数组的第一个数组元素的值
  2. 每一次循环，把返回值给做为 下一次循环的上一次值
  3. 如果有起始值，则 起始值做为上一次值

```
arr.reduce(function(上一次返回值, 当前数组元素){}, 初始值)
```



总结

方法	作用	说明
forEach	遍历数组	不返回数组，经常用于查找遍历数组元素
map	迭代数组	返回新数组，返回的是处理之后的数组元素，想要使用返回的新数组
join	拼接数组元素	返回指定连接符拼接得到的字符串
reduce	累计器	返回累计处理的结果，经常用于求和等

练习

计算薪资案例

```
arr.reduce(function(上一次返回值, 当前数组元素){}, 初始值)
```

需求：

①：根据数据计算当月支出薪资

数据：

```
const arr = [{
  name: '张三',
  salary: 10000
}, {
  name: '李四',
  salary: 15000
}, {
  name: '王五',
  salary: 20000
}
]
```

目录

Contents

案例

购物车展示

需求：展示购物车列表数据

	称心如意手摇咖啡磨豆机咖啡豆研磨机	¥289.9	x2	¥579.8
	竹制干泡茶盘正方形沥水茶台品茶盘	¥109.8	x3	¥329.4
	古法温酒汝瓷酒具套装白酒杯莲花温酒器	¥488	x1	¥488
	大师监制龙泉青瓷茶叶罐	¥139	x2	¥278

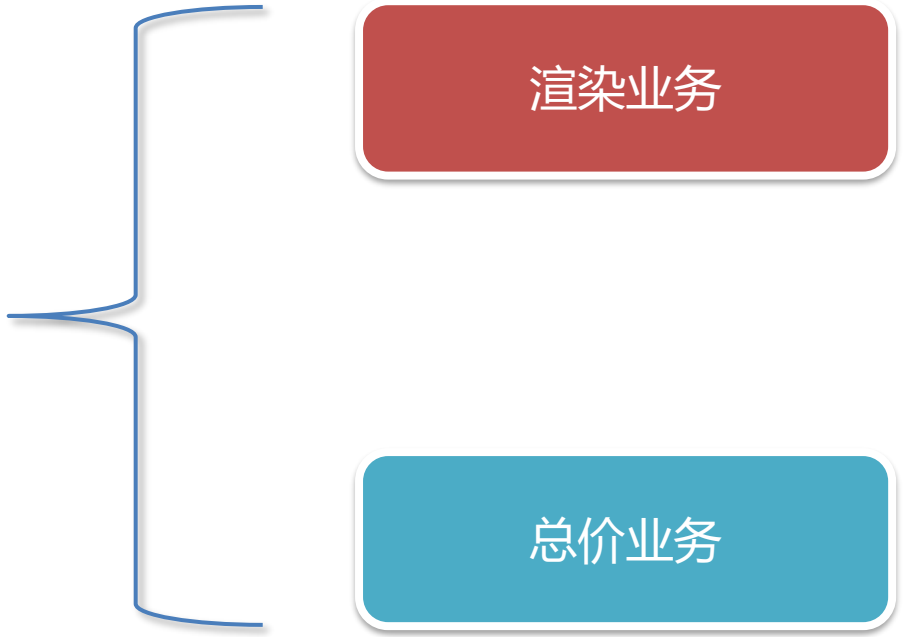
合计：¥1675.20

案例 购物车展示

分析业务模块：

	称心如意手摇咖啡磨豆机咖啡豆研磨机	¥289.9	x2	¥579.8
	竹制干泡茶盘正方形沥水茶台品茶盘	¥109.8	x3	¥329.4
	古法温酒汝瓷酒具套装白酒杯莲花温酒器	¥488	x1	¥488
	大师监制龙泉青瓷茶叶罐	¥139	x2	¥278

合计：¥1675.20



步骤 购物车展示

渲染业务

①：渲染业务

先利用 map+join 来遍历，有多少条数据，渲染多少商品

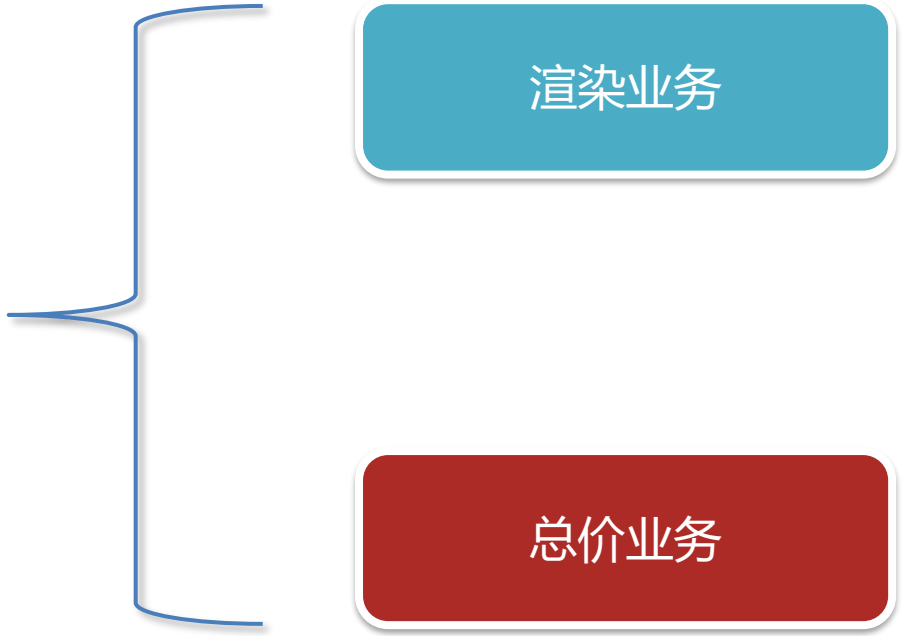
- 更换数据：图片，商品名称，单价，数量
- 采取对象解构的方式

案例 购物车展示

分析业务模块：

	称心如意手摇咖啡磨豆机咖啡豆研磨机	¥289.9	x2	¥579.8
	竹制干泡茶盘正方形沥水茶台品茶盘	¥109.8	x3	¥329.4
	古法温酒汝瓷酒具套装白酒杯莲花温酒器	¥488	x1	¥488
	大师监制龙泉青瓷茶叶罐	¥139	x2	¥278

合计：¥1675.20



步骤 购物车展示

总价业务

②：总价业务

- 求和用到数组 reduce 方法 累计器， 总价: total
- 根据数据里面的数量和单价累加和即可
- 注意 reduce方法有2个参数，第一个是回调函数，第二个是 初始值，这里写 0
- 注意 总价要保留2位小数 toFixed(2)

传智教育旗下高端IT教育品牌