JavaScript 进阶第二天

构造函数&数据常用函数

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

1. 掌握基于构造函数创建对象，理解实例化过程
2. 掌握对象数组字符数字等类型的常见属性和方法，便捷完成功能

目录

Contents

深入对象

- 创建对象三种方式
- 构造函数
- 实例成员&静态成员

1.1创建对象三种方式

目标：了解创建对象有三种方式

1. **利用对象字面量创建对象**

```
const o = {
  name: '佩奇'
}
```

2. **利用 new Object 创建对象**

```
const o = new Object({ name: '佩奇' })
console.log(o)  // {name: '佩奇'}
```

3. **利用构造函数创建对象**

深入对象

- 创建对象三种方式
- 构造函数
- 实例成员&静态成员

1.2 构造函数

目标：能够利用构造函数创建对象

- **构造函数：**是一种特殊的函数，主要用来初始化对象
- **使用场景：**常规的 {...} 语法允许创建一个对象。比如我们创建了佩奇的对象，继续创建乔治的对象还需要重新写一遍，此时可以通过构造函数来快速创建多个类似的对象。

```
// 创建佩奇的对象
const Peppa = {
  name: '佩奇',
  age: 6,
  gender: '女'
}
```

```
// 创建乔治的对象
const George = {
  name: '乔治',
  age: 3,
  gender: '男'
}
```

```
// 创建猪妈妈的对象
const Mum = {
  name: '猪妈妈',
  age: 30,
  gender: '女'
}
```

```
// 创建猪爸爸的对象
const Dad = {
  name: '猪爸爸',
  age: 32,
  gender: '男'
}
```

```
function Pig(name, age, gender) {
  this.name = name
  this.age = age
  this.gener = gender
}
// 创建佩奇对象
const Peppa = new Pig('佩奇', 6, '女')
// 创建乔治对象
const George = new Pig('乔治', 3, '男')
// 创建猪妈妈对象
const Mum = new Pig('猪妈妈', 30, '女')
// 创建猪爸爸对象
const Dad = new Pig('猪爸爸', 32, '男')
console.log(Peppa) // {name: '佩奇', age: 6, gener: '女'}
```

1.2 构造函数

构造函数在技术上是常规函数。

不过有两个约定：

1. 它们的命名以大写字母开头。
2. 它们只能由 "new" 操作符来执行。

```
function Pig(name, age, gender) {
  this.name = name
  this.age = age
  this.gener = gender
}
// 创建佩奇对象
const Peppa = new Pig('佩奇', 6, '女')
// 创建乔治对象
const George = new Pig('乔治', 3, '男')
// 创建猪妈妈对象
const Mum = new Pig('猪妈妈', 30, '女')
// 创建猪爸爸对象
const Dad = new Pig('猪爸爸', 32, '男')
console.log(Peppa) // {name: '佩奇', age: 6, gener: '女'}
```

1.2 构造函数

- **构造函数语法：**大写字母开头的函数
- **创建构造函数：**

```
// 1. 创建构造函数
function Pig(name) {
  this.name = name
}
// 2. new 关键字调用函数
// new Pig('佩奇')
// 接受创建的对象
const peppa = new Pig('佩奇')
console.log(peppa) // {name: '佩奇'}
```

说明：

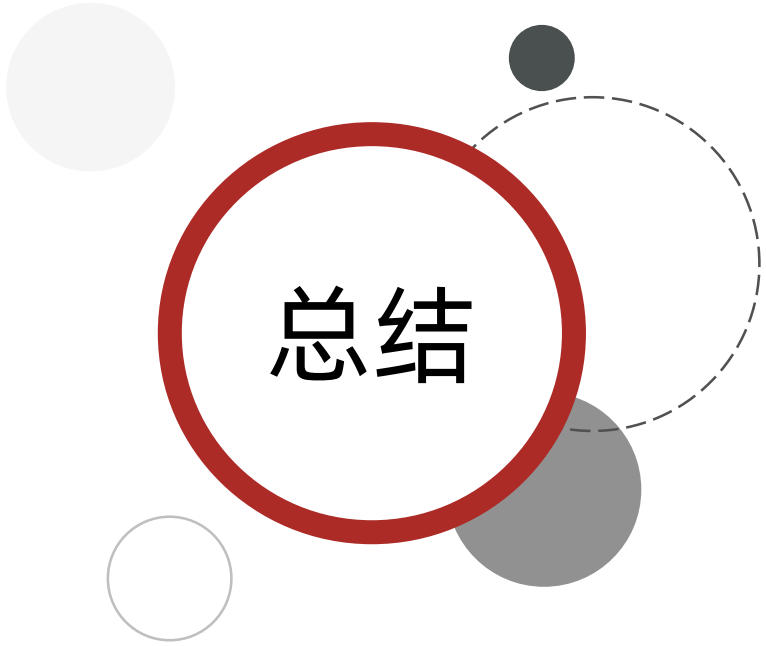
1. 使用 new 关键字调用函数的行为被称为实例化

2. 实例化构造函数时没有参数时可以省略 ()

3. 构造函数内部无需写return，返回值即为新创建的对象

4. 构造函数内部的 return 返回的值无效，所以不要写return

5. new Object（） new Date（） 也是实例化构造函数



1. 构造函数的作用是什么？怎么写呢？
   - 构造函数是来快速创建多个类似的对象
   - 大写字母开头的函数
2. new 关键字调用函数的行为被称为？
   - 实例化
3. 构造函数内部需要写return吗，返回值是什么？
   - 不需要
   - 构造函数自动返回创建的新的对象

练习

- **利用构造函数创建多个对象**

需求：

①：写一个Goods构造函数

②：里面包含属性 name 商品名称 price 价格 count 库存数量

③：实例化多个商品对象，并打印到控制台，例如

小米 1999 20

华为 3999 59

vivo 1888 100

1.2 构造函数

- **实例化执行过程**

说明：

1. 创建新对象

2. 构造函数this指向新对象

3. 执行构造函数代码，修改this，添加新的属性

4. 返回新对象

```
// 1. 创建构造函数
function Pig(name) {
  this.name = name
}
// 2. new 关键字调用函数
// new Pig('佩奇')
// 接受创建的对象
const peppa = new Pig('佩奇')
console.log(peppa) // {name: '佩奇'}
```

深入对象

- 创建对象三种方式
- 构造函数
- 实例成员&静态成员

1.3实例成员&静态成员

目标：能够说出什么是实例成员和静态成员

实例成员：

通过构造函数创建的对象称为实例对象，实例对象中的属性和方法称为实例成员。

```
// 构造函数
function Person() {
  // 构造函数内部的 this 就是实例对象
  // 实例对象中动态添加属性
  this.name = '小明'
  // 实例对象动态添加方法
  this.sayHi = function () {
    console.log('大家好~')
  }
}
// 实例化，p1 是实例对象
// p1 实际就是 构造函数内部的 this
const p1 = new Person()
console.log(p1)
console.log(p1.name) // 访问实例属性
p1.sayHi() // 调用实例方法
```

说明：

1. 实例对象的属性和方法即为实例成员
2. 为构造函数传入参数，动态创建结构相同但值不同的对象
3. 构造函数创建的实例对象彼此独立互不影响。

1.3实例成员&静态成员

目标：能够说出什么是实例成员和静态成员

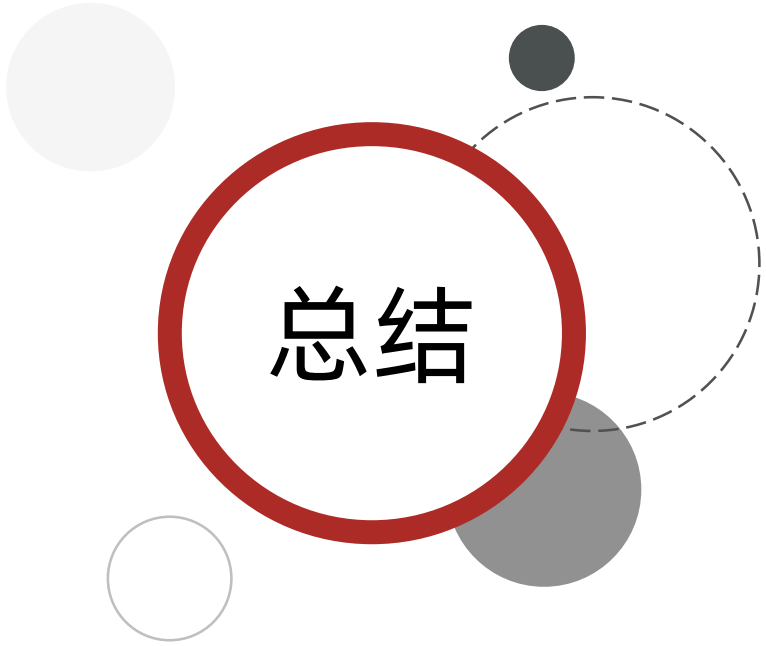
静态成员：

构造函数的属性和方法被称为静态成员

```
// 构造函数
function Person(name, age) {
  // 省略实例成员
}
// 静态属性
Person.eyes = 2
Person.arms = 2
// 静态方法
Person.walk = function () {
  console.log('^_^人都会走路...')
  // this 指向 Person
  console.log(this.eyes)
}
```

说明：

1. 构造函数的属性和方法被称为静态成员
2. 一般公共特征的属性或方法静态成员设置为静态成员
3. 静态成员方法中的 this 指向构造函数本身



1. 什么是实例成员？
   - 实例对象的属性和方法即为实例成员
2. 什么是静态成员？
   - 构造函数的属性和方法被称为静态成员

目录

Contents

内置构造函数

- Object
- Array
- String
- Number

2. 内置构造函数

在 JavaScript 中最主要的数据类型有 6 种：

基本数据类型：

- 字符串、数值、布尔、undefined、null

引用类型：

- 对象

但是，我们会发现有些特殊情况：

```
// 普通字符串
const str = 'andy'
console.log(str.length) // 4
```

其实字符串、数值、布尔、等基本类型也都有专门的构造函数，这些我们称为包装类型。

JS中几乎所有的数据都可以基于构成函数创建。

2. 内置构造函数

引用类型

- Object，Array，RegExp，Date 等

包装类型

- String，Number，Boolean 等

内置构造函数

- Object
- Array
- String
- Number

2.1 Object

Object 是内置的构造函数，用于创建普通对象。

```
// 通过构造函数创建普通对象
const user = new Object({name: '小明', age: 15})
```

推荐使用字面量方式声明对象，而不是 Object 构造函数

2.1 Object

学习三个常用静态方法（静态方法就是只有构造函数Object可以调用的）

```
// 想要获得对象里面的属性和值怎么做的？
const o = { name: '佩奇', age: 6 }
```

```
for (let k in o) {
  console.log(k) // 属性  name  age
  console.log(o[k]) // 值  佩奇  6
}
```

现在有新的方法了~~~~

2.1 Object

学习三个常用静态方法（静态方法就是只有构造函数Object可以调用的）

- **作用：** Object.keys 静态方法获取对象中所有属性（键）
- **语法：**

```
const o = { name: '佩奇', age: 6 }
// 获得对象的所有键，并且返回是一个数组
const arr = Object.keys(o)
console.log(arr) // ['name', 'age']
```

- **注意：** 返回的是一个数组

2.1 Object

学习三个常用静态方法（静态方法就是只有构造函数Object可以调用的）

- **作用：** Object.values 静态方法获取对象中所有属性值
- **语法：**

```
const o = { name: '佩奇', age: 6 }
// 获得对象的所有值，并且返回是一个数组
const arr = Object.values(o)
console.log(arr) // ['佩奇', 6]
```

- **注意：** 返回的是一个数组

2.1 Object

学习三个常用静态方法（静态方法就是只有构造函数Object可以调用的）

- **作用：** Object. assign 静态方法常用于对象拷贝
- **语法：**

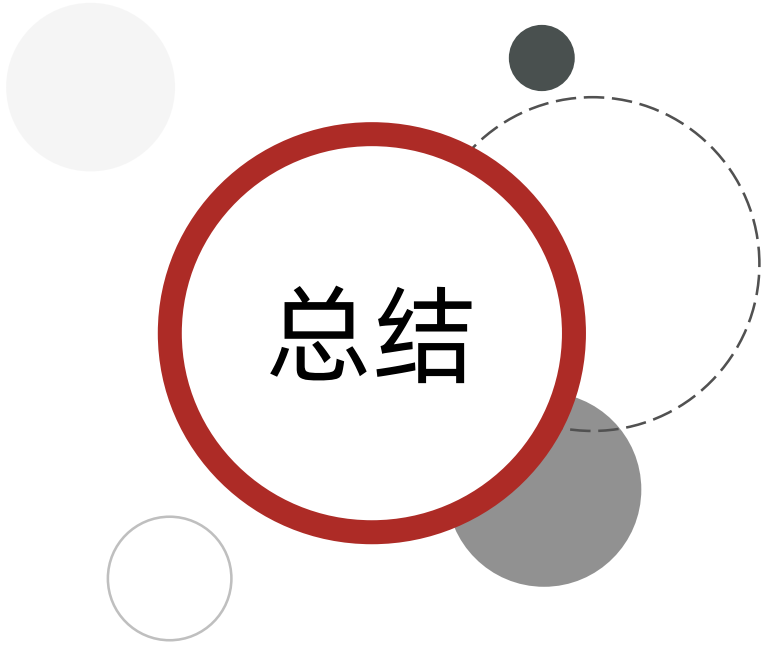
```
// 拷贝对象 把 o 拷贝给 obj
const o = { name: '佩奇', age: 6 }
const obj = {}
Object.assign(obj, o)
console.log(obj) // {name: '佩奇', age: 6}
```

2.1 Object

学习三个常用静态方法（静态方法就是只有构造函数Object可以调用的）

- **作用：** Object. assign 静态方法常用于对象拷贝
- **使用：** 经常使用的场景给对象添加属性

```
// 给 o 新增属性
const o = { name: '佩奇', age: 6 }
Object.assign(o, { gender: '女' })
console.log(o) //{name: '佩奇', age: 6, gender: '女'}
```



1. 什么是静态方法?
   - 只能给构造函数使用的方法 比如 Object.keys()
2. Object.keys()方法的作用是什么？
   - 获取对象中所有属性（键）
3. Object.values()方法的作用是什么？
   - 获取对象中所有属性值（值）

内置构造函数

- Object
- Array
- String
- Number

2.2 Array

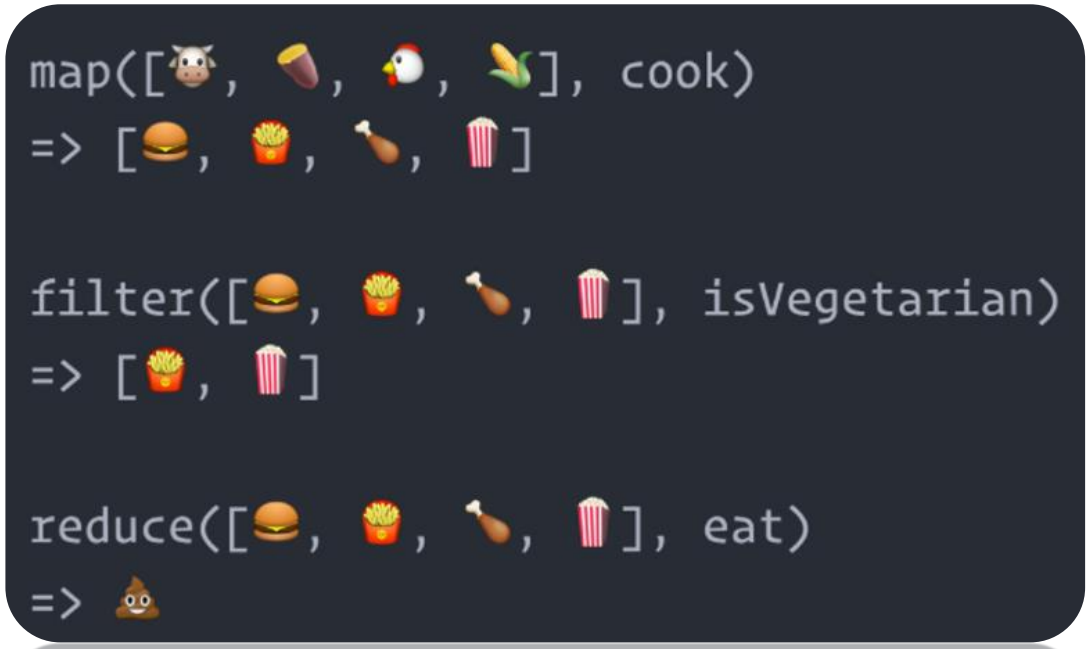
Array 是内置的构造函数，用于创建数组

```
const arr = new Array(3, 5)
console.log(arr) // [3,5]
```

创建数组建议使用字面量创建，不用 Array构造函数创建

2.2 Array

1. 数组常见实例方法-核心方法



方法	作用	说明
forEach	遍历数组	不返回，用于不改变值，经常用于查找打印输出值
filter	过滤数组	筛选数组元素，并生成新数组
map	迭代数组	返回新数组，新数组里面的元素是处理之后的值，经常用于处理数据
reduce	累计器	返回函数累计处理的结果，经常用于求和等

2.2 Array

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

```
arr.reduce(function(){}, 起始值)
```

- **参数：**

  起始值可以省略，如果写就作为第一次累计的起始值

2.2 Array

- **作用：**reduce 返回函数累计处理的结果，经常用于求和等
- **语法：**

```
arr.reduce(function(累计值， 当前元素 [,索引号][,源数组]){}， 起始值)
```

- **累计值参数：**

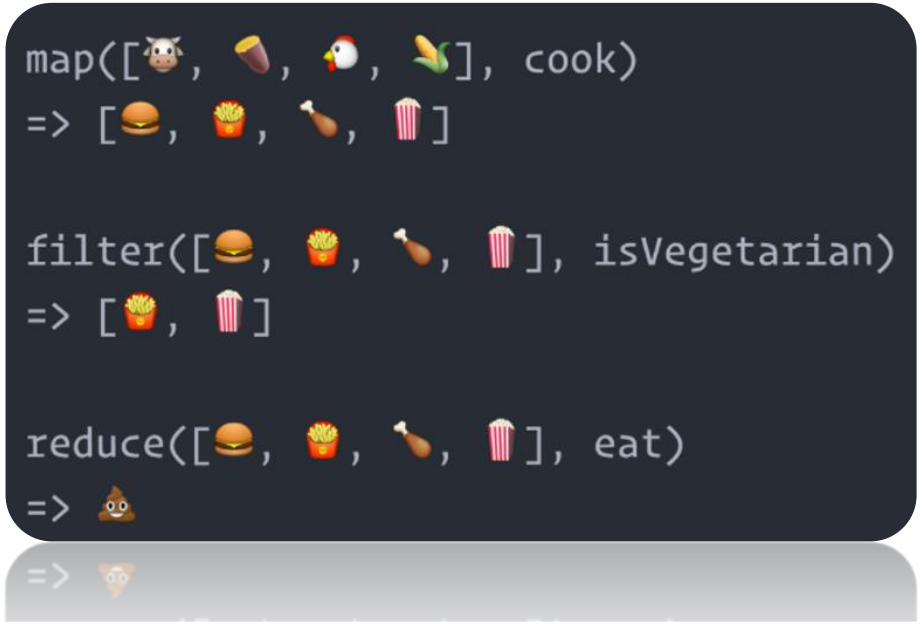
1. 如果有起始值，则以起始值为准开始累计， 累计值 = 起始值
2. 如果没有起始值， 则累计值以数组的第一个数组元素作为起始值开始累计
3. 后面每次遍历就会用后面的数组元素 累计到 累计值 里面 （类似求和里面的 sum ）

2.2 Array

- **作用：**reduce 返回函数累计处理的结果，经常用于求和等
- **使用场景：求和运算：**

```
const arr = [1, 5, 9]
const count = arr.reduce((prev, item) => prev + item)
console.log(count)
```

总结



方法	作用	说明
forEach	遍历数组	不返回，用于不改变值，经常用于查找打印输出值
filter	过滤数组	筛选数组元素，并生成新数组
map	迭代数组	返回新数组，新数组里面的元素是处理之后的值，经常用于处理数据
reduce	累计器	返回函数累计处理的结果，经常用于求和等

练习

- **员工涨薪计算成本**

需求：

①：给员工每人涨薪 30%

②：然后计算需要支出的费用

数据：

```
// 第一个参数是回调函数
// 第二参数如果不写，默认是prev 第一个值
reduce(callback, 传递给函数的初始值)
```

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

练习

- **员工涨薪计算成本**

答案：

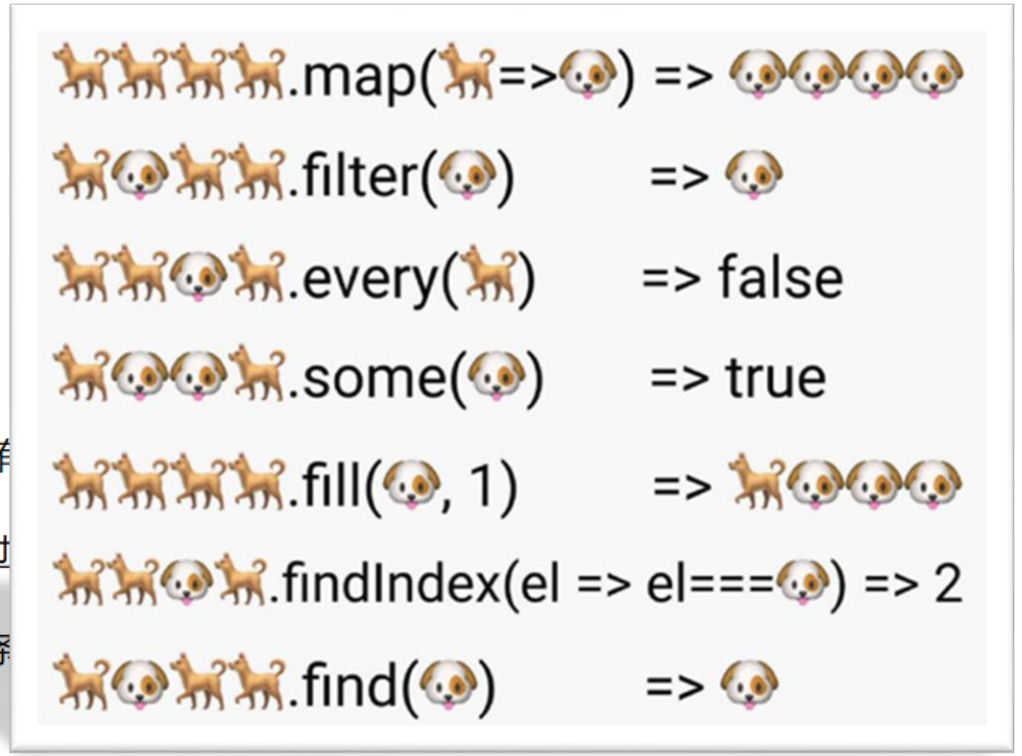
```
const arr = [{
  name: '张三',
  salary: 10000
}, {
  name: '李四',
  salary: 10000
}, {
  name: '王五',
  salary: 20000
},
]

const total = arr.reduce((prev, item) => prev += item.salary * 0.3, 0)
console.log(total)
```

2.2 Array

2. 数组常见方法-其他方法

5. 实例方法 `join` 数组元素拼接为字符串，返回字符串(重点)
6. 实例方法 `find` 查找元素， 返回符合测试条件的第一个数组元素值，如果没有
7. 实例方法 `every` 检测数组所有元素是否都符合指定条件，如果**所有元素**都通过
8. 实例方法 `some` 检测数组中的元素是否满足指定条件 **如果数组中有**元素满足条
9. 实例方法 `concat` 合并两个数组，返回生成新数组
10. 实例方法 `sort` 对原数组单元值排序
11. 实例方法 `splice` 删除或替换原数组单元
12. 实例方法 `reverse` 反转数组
13. 实例方法 `findIndex` 查找元素的索引值



练习


温柔甜美，女童基础针织开衫3-12岁
微收版型，舒适不紧勒
¥89.00 ¥89.00
促销 12月好物放送，App领券购买直降120元
配送 至 北京市 市辖区 东城区
服务 •无忧退货 •快速退款 •免费包邮 了解详情
颜色
尺码
160cm
150cm
120cm
110cm
140cm
130cm
颜色
浅麻灰
蔚兰色
花杏色
姜黄
珍珠粉
尺码
160cm
150cm
120cm
110cm
140cm
130cm
数量
-
1
+
加入购物车


全选
商品信息
单价
温柔甜美，女童基础针织开衫3-12岁
颜色:珍珠粉 尺码:160cm 颜色:珍珠...
¥89.00
<span class="ellipsis" data-v-f7d5aa52>
颜色:珍珠粉 尺码:160cm 颜色:珍珠粉 尺
码:160cm == $0

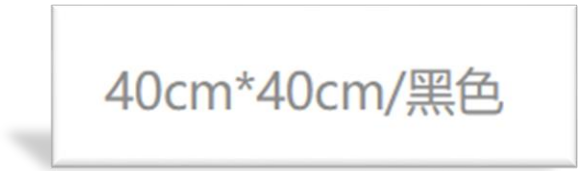
练习

```
const goodsList = [
  {
    id: '4001172',
    name: '称心如意手摇咖啡磨豆机咖啡豆研磨机',
    price: 289.9,
    picture: 'https://yanxuan-item.nosdn.127.net/84a59ff9c58a77032564e61f716846d6.jpg',
    count: 2,
    spec: { color: '白色' }
  },
  {
    id: '4001009',
    name: '竹制干泡茶盘正方形沥水茶台品茶盘',
    price: 109.8,
    picture: 'https://yanxuan-item.nosdn.127.net/2d942d6bc94f1e230763e1a5a3b379e1.png',
    count: 3,
    spec: { size: '40cm*40cm' , color: '黑色'}
  },
  {
    id: '4001874',
    name: '古法温酒汝瓷酒具套装白酒杯莲花温酒器',
    price: 488,
    picture: 'https://yanxuan-item.nosdn.127.net/44e51622800e4fceb6bee8e616da85fd.png',
    count: 1,
    spec: { color: '青色', sum: '一大四小' }
  },
```

练习 • **请完成以下需求**

const spec = { size: '40cm*40cm' , color: '黑色'}

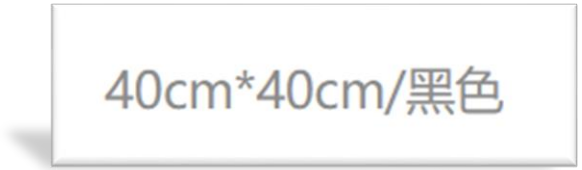
请将里面的值写到div标签里面，展示内容如下：



练习 • 请完成以下需求

const spec = { size: '40cm*40cm' , color: '黑色'}

请将里面的值写到div标签里面，展示内容如下：



思路： 获得所有的属性值，然后拼接字符串就可以了

①： 获得所有属性值是： Object.values() 返回的是数组

②： 拼接数组是 join('') 这样就可以转换为字符串了

```
const spec = { size: '40cm*40cm', color: '黑色' }
// console.log(Object.values(spec))
document.querySelector('div').innerHTML = Object.values(spec).join('/')
// 第一种方法更简单，万一对象里面有多个属性，第二种方法就不方便了
// document.querySelector('div').innerHTML = spec.size + '/' + spec.color
```

2.2 Array

2. 数组常见方法- 伪数组转换为真数组

静态方法 Array.from()

内置构造函数

- Object
- Array
- String
- Number

2.3 String

在 JavaScript 中的字符串、数值、布尔具有对象的使用特征，如具有属性和方法

```
// 字符串类型
const str = 'hello world!'
  // 统计字符的长度（字符数量）
console.log(str.length)

// 数值类型
const price = 12.345
// 保留两位小数
price.toFixed(2)
```

之所以具有对象特征的原因是字符串、数值、布尔类型数据是 JavaScript 底层使用 Object 构造函数“包装”来的，被称为**包装类型**。

2.3 String

1. 常见实例方法

1. 实例属性 `length` 用来获取字符串的度长(重点)
2. 实例方法 `split('分隔符')` 用来将字符串拆分成数组(重点)
3. 实例方法 `substring（需要截取的第一个字符的索引[,结束的索引号]）` 用于字符串截取(重点)
4. 实例方法 `startsWith(检测字符串[, 检测位置索引号])` 检测是否以某字符开头(重点)
5. 实例方法 `includes(搜索的字符串[, 检测位置索引号])` 判断一个字符串是否包含在另一个字符串中，根据情况返回 true 或 false(重点)
6. 实例方法 `toUpperCase` 用于将字母转换成大写
7. 实例方法 `toLowerCase` 用于将就转换成小写
8. 实例方法 `indexOf` 检测是否包含某字符
9. 实例方法 `endsWith` 检测是否以某字符结尾
10. 实例方法 `replace` 用于替换字符串，支持正则匹配
11. 实例方法 `match` 用于查找字符串，支持正则匹配

练习

- **请完成以下需求**

```
{
  id: '4001649',
  name: '大师监制龙泉青瓷茶叶罐',
  price: 139,
  picture: 'https://yanxuan-item.nosdn.127.net/4356c9fc150753775fe56b465314f1eb.png',
  count: 1,
  spec: { size: '小号', color: '紫色' },
  gift: '50g茶叶,清洗球'
}
```

大师监制龙泉青瓷茶叶罐

小号/紫色

【赠品】50g茶叶

【赠品】清洗球

练习 · 显示赠品练习

请将下面字符串渲染到准备好的 p标签内部，结构必须如左下图所示，展示效果如右图所示：

const gift = '50g茶叶,清洗球'

```
<p class="name"> == $0
  <span class="tag">【赠品】50g茶叶</span>
  <span class="tag">【赠品】清洗球</span>
</p>
```

【赠品】50g茶叶
【赠品】清洗球

思路：

①：把字符串拆分为数组，这样两个赠品就拆分开了 用那个方法？ split(‘,’)

②：利用map遍历数组，同时把数组元素生成到span里面，并且返回

③：因为返回的是数组，所以需要 转换为字符串, 用那个方法？ join(‘’)

④：p的innerHTML 存放刚才的返回值

练习 · 显示赠品练习

请将下面字符串渲染到准备好的 p标签内部，结构必须如左下图所示，展示效果如右图所示：

【赠品】50g茶叶
【赠品】清洗球

```
// 答案
const p = document.querySelector('.name')
const gift = '50g茶叶,清洗球'
p.innerHTML = gift.split(',').map(item => `<span class='tag'>【赠品】${item}</span> <br>`).join('')
```

内置构造函数

- Object
- Array
- String
- Number

2.4 Number

Number 是内置的构造函数，用于创建数值

常用方法：

toFixed() 设置保留小数位的长度

```
// 数值类型
const price = 12.345
// 保留两位小数 四舍五入
console.log(price.toFixed(2))   // 12.35
```

目录

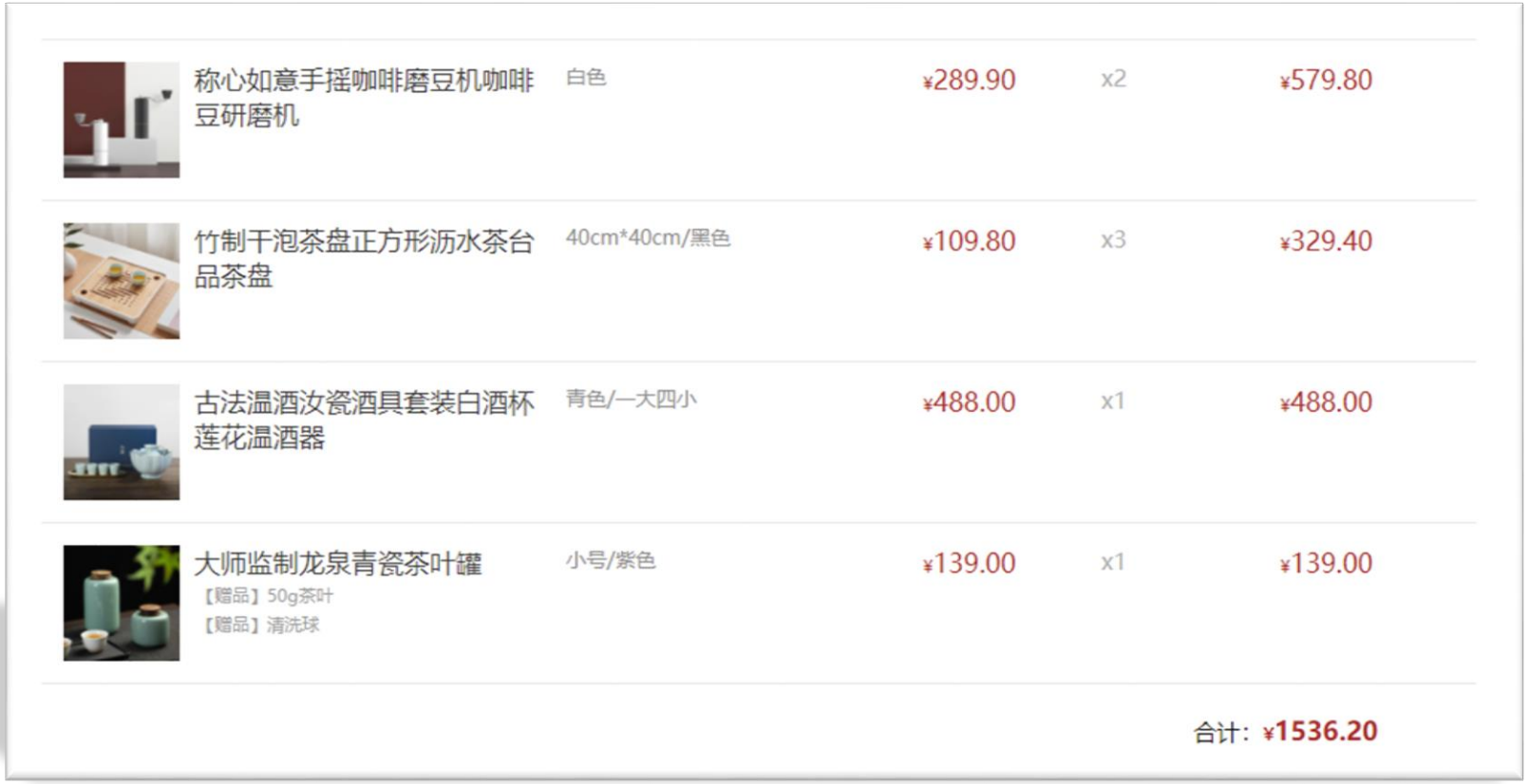
Contents

案例

购物车展示

需求：

根据后台提供的数据，渲染购物车页面



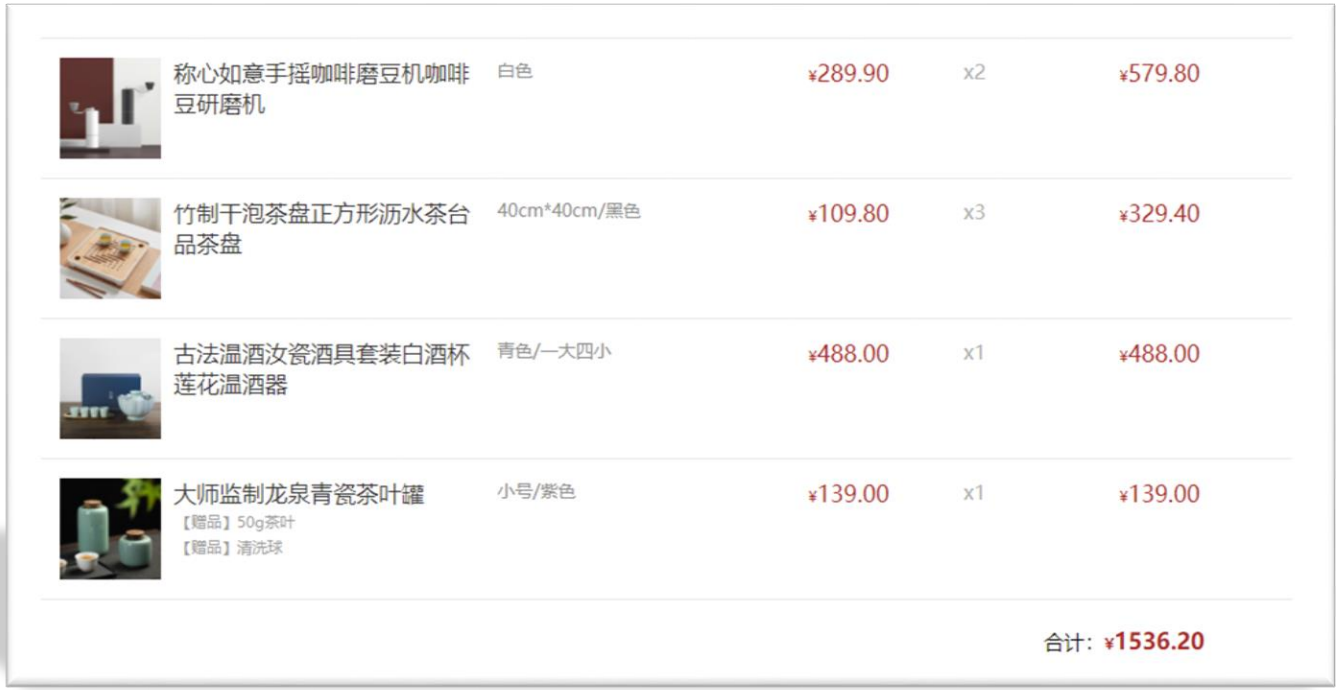
案例 购物车展示

分析业务模块：

①：渲染图片、标题、颜色、价格、赠品等数据

②：单价和小计模块

③：总价模块



步骤 • 购物车展示

分析业务模块：

①：把整体的结构直接生成然后渲染到大盒子.list 里面

②：那个方法可以遍历的同时还有返回值 map 方法

③：最后计算总价模块，那个方法可以求和？reduce 方法

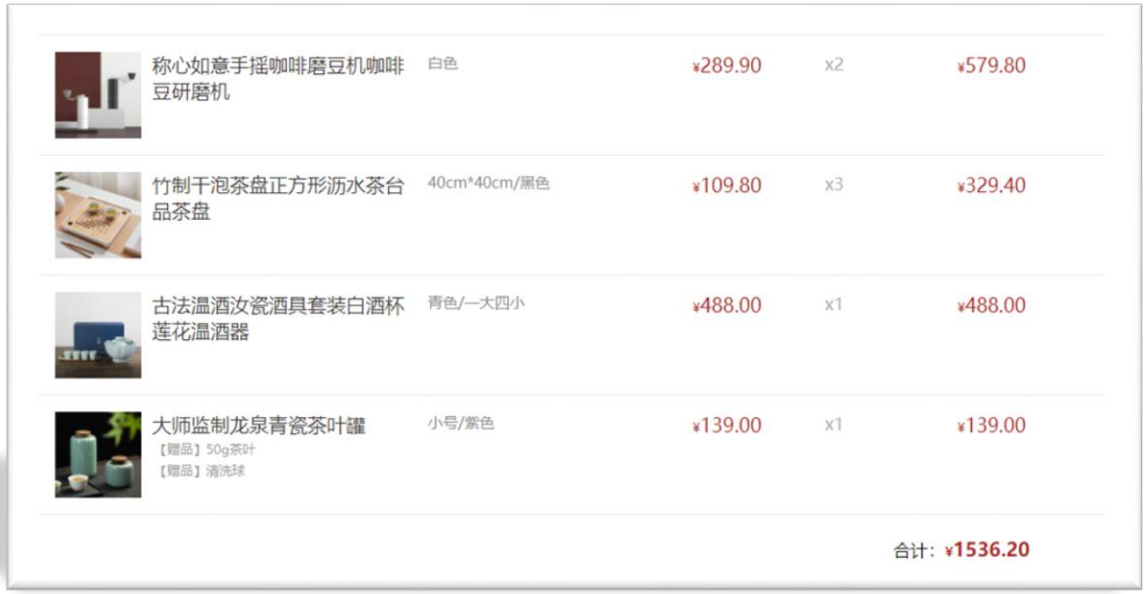
步骤 • 购物车展示

分析业务模块：

①：先利用map来遍历，有多少条数据，渲染多少相同商品

②：里面更换各种数据，注意使用对象解构赋值

③：利用reduce计算总价

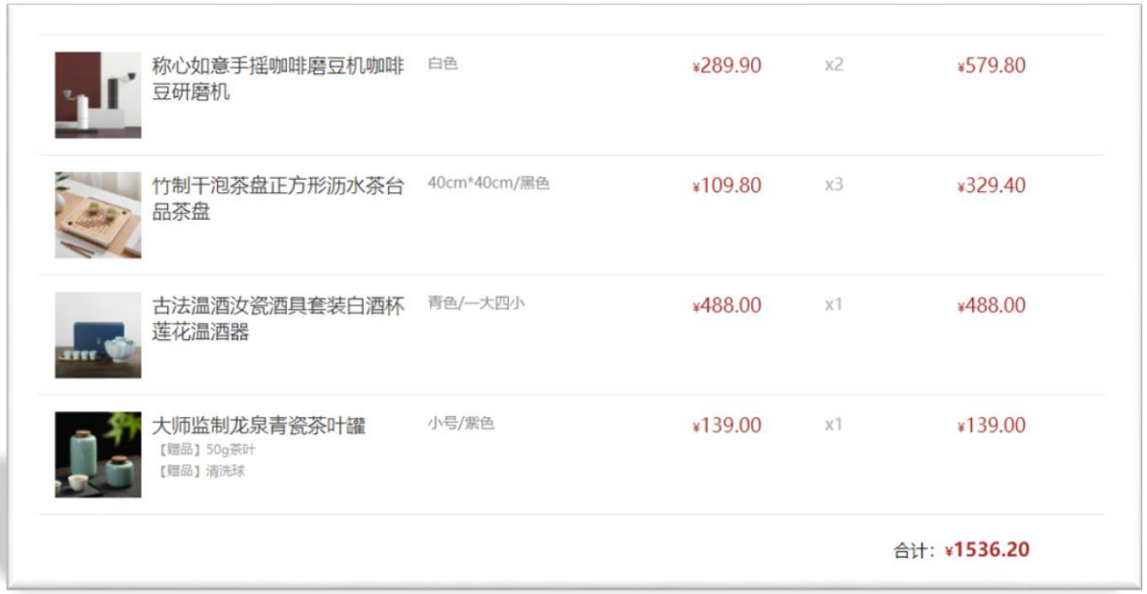


步骤 • 购物车展示

分析业务模块：

①：先利用map来遍历，有多少条数据，渲染多少相同商品

- 可以先写死的数据
- 注意map返回值是 数组，我们需要用 join 转换为字符串
- 把返回的字符串 赋值 给 list 大盒子的 innerHTML

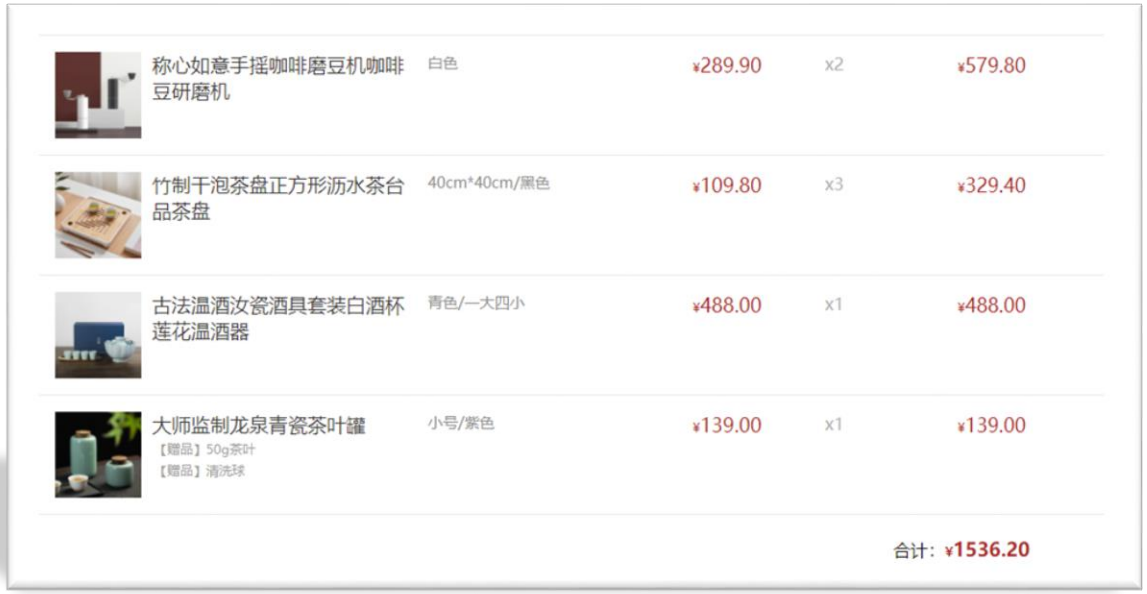


步骤 • 购物车展示

分析业务模块：

②：更换数据

- 先更换不需要处理的数据，图片，商品名称，单价，数量
- 采取对象解构的方式
- 注意 单价要保留2位小数， 489.00 toFixed(2)

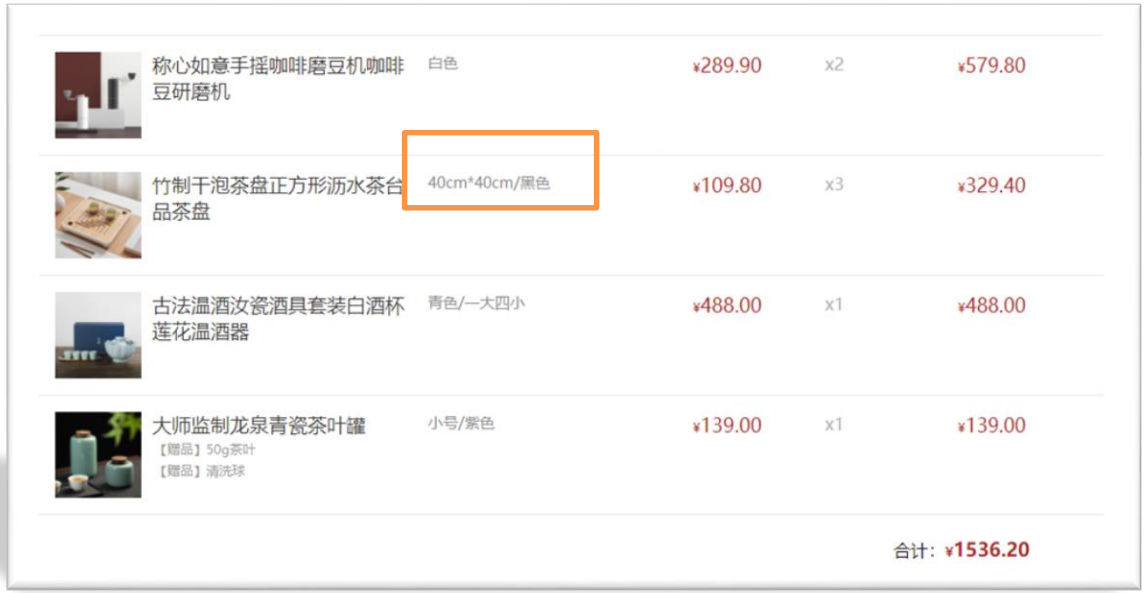


步骤 • 购物车展示

分析业务模块：

②：更换数据 - 处理 规格文字 模块

- 获取 每个对象里面的 spec，上面对象解构添加 spec
- 获得所有属性值是： Object.values() 返回的是数组
- 拼接数组是 join('') 这样就可以转换为字符串了




```
{
  id: '4001649',
  name: '大师监制龙泉青瓷茶叶罐',
  price: 139,
  picture: 'https://yanxuan-item.nosdn.127.net/4356c9
  count: 1,
  spec: { size: '小号', color: '紫色' },
  gift: '50g茶叶,清洗球'
}
```

步骤 • 购物车展示

分析业务模块：

②：更换数据 - 处理 赠品 模块

- 获取 每个对象里面的 gift，上面对象解构添加 gift

思路：

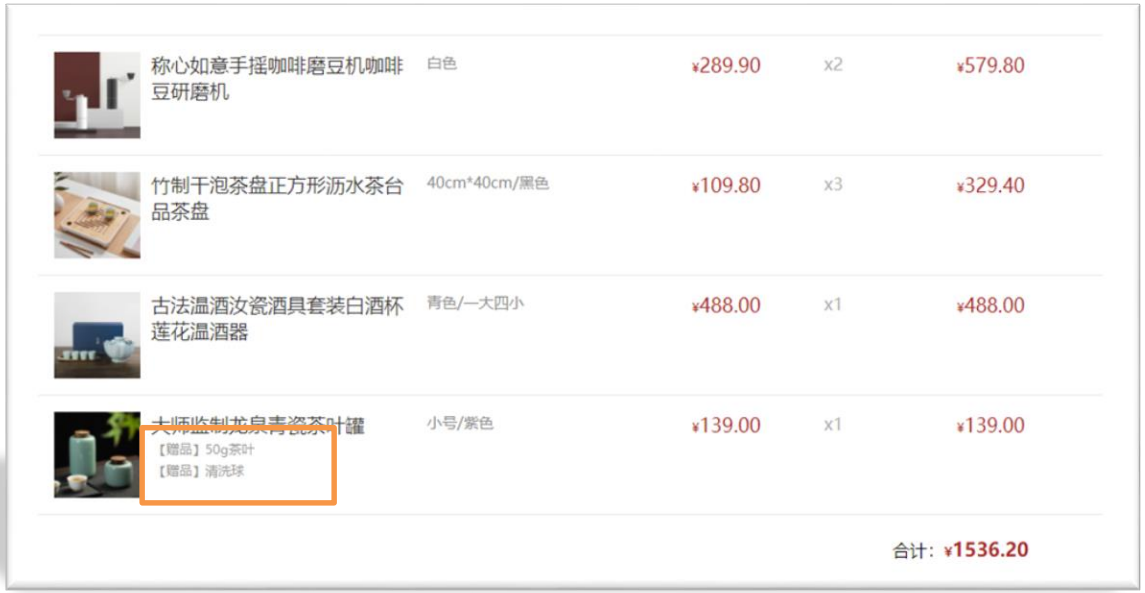
①：把字符串拆分为数组，这样两个赠品就拆分开了 用那个方法？ split(',')

②：利用map遍历数组，同时把数组元素生成到span里面，并且返回

③：因为返回的是数组，所以需要 转换为字符串，用那个方法？ join('')

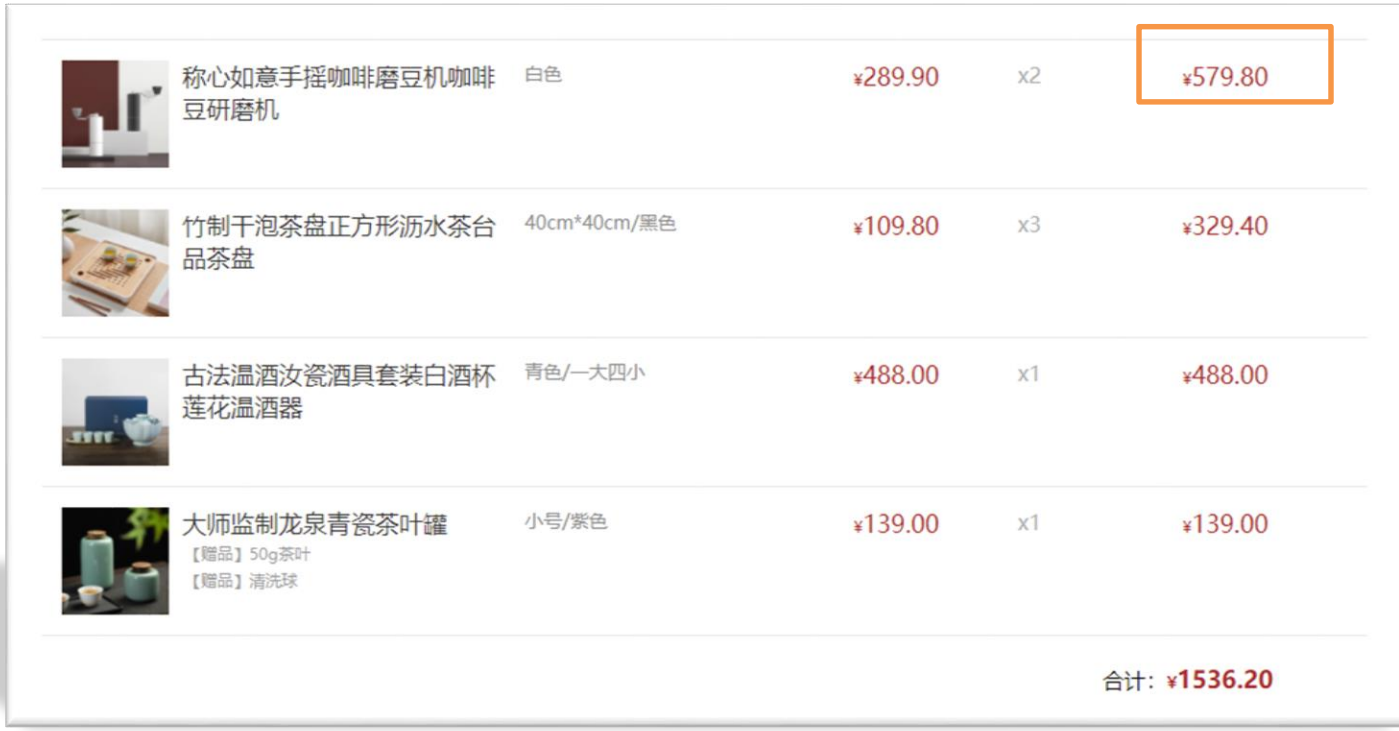
注意要判断是否有gif属性，没有的话不需要渲染

利用变成的字符串然后写到 p.name里面




```
{
  id: '4001649',
  name: '大师监制龙泉青瓷茶叶罐',
  price: 139,
  picture: 'https://yanxuan-item.nosdn.127.net/435
  count: 1,
  spec: { size: '小号', color: '紫色' },
  gift: '50g茶叶,清洗球'
}
```

步骤 • 购物车展示



分析业务模块：

②：更换数据 - 处理 小计 模块

- 小计 = 单价 * 数量
- 小计名可以为：subTotal = price * count
- 注意保留2位小数

关于小数的计算精度问题：

0.1 + 0.2 = ?

解决方案： 我们经常转换为整数

（0.1*100 + 0.2*100）/ 100 === 0.3

这里是给大家拓展思路和处理方案

步骤 • 购物车展示

分析业务模块：

③：计算 合计 模块

- 求和用到数组 reduce 方法 累计器
- 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

今日作业

1. 整理笔记
2. 综合案例至少写三遍
3. 开始做测试题： PC端地址： https://ks.wjx.top/vj/QG54lSj.aspx
4. 明天上午总结这2天内容（重点是笔记和综合案例）
5. 明天下午预习第三天内容

传智教育旗下高端IT教育品牌