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
- 内置构造函数
- 综合案例

深入对象

- 创建对象三种方式
- 构造函数
- 实例成员&静态成员
- 一切皆对象

1.1创建对象三种方式（导师讲解）

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
- 一切皆对象

1.2 构造函数

- **构造函数 ：**是一种特殊的函数，主要用来创建对象(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. 通过 new 关键字来调用构造函数，可以创建对象

```
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'}
```

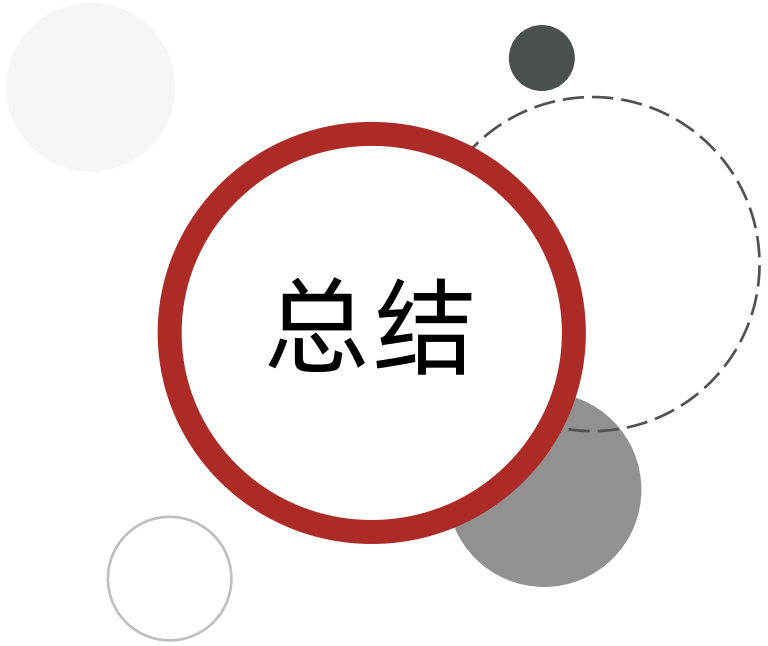
1.2 构造函数

- **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. 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

- **new 实例化执行过程**

1. 创建新空对象
2. 构造函数this指向新对象
3. 执行构造函数代码
4. 返回新对象

```
function Pig(name, age, gender) {
    this.name = name
    this.age = age
    this.gender = gender
}
const peiqi = new Pig('佩奇', 6, '女')
console.log(peiqi)
```

```
name: ''佩奇',
age: 6,
gender: '女'
```

深入对象

- 创建对象三种方式
- 构造函数
- 实例成员&静态成员
- 一切皆对象

1.3 实例成员&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（实例属性和实例方法）

```
// 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. 为构造函数传入参数，创建结构相同但值不同的对象
2. 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

静态成员：

构造函数的属性和方法被称为静态成员（静态属性和静态方法）

```
// 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. 静态成员只能构造函数来访问
2. 静态方法中的this指向构造函数

比如 Date.now()　　Math.PI　　Math.random()

总结

1. 实例成员(属性和方法）写在谁身上?
   - 实例对象的属性和方法即为实例成员
   - 实例对象相互独立，实例成员当前实例对象使用
2. 静态成员(属性和方法）写在谁身上?
   - 构造函数的属性和方法被称为静态成员
   - 静态成员只能构造函数访问

深入对象

- 创建对象三种方式
- 构造函数
- 实例成员&静态成员
- 一切皆对象

1.4 一切皆对象

引用类型：

- Object，Array，RegExp，Date 等

基本数据类型：

- 字符串、数值、布尔、undefined、null

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

1.4 一切皆对象

包装类型

- String，Number，Boolean

包装类型执行过程：

- 创建一个 String 类型的实例
- 调用实例上的特定方法
- 销毁实例

JS中几乎所有的数据都可以基于构造函数创建，不同的构造器创建出来的数据拥有不同的属性和方法

总结

1. 引用数据类型属于对象吗？
   - 属于，数组，函数，正则，日期都属于对象
2. 为什么基本数据类型比如字符串会有属性和方法？
   - 因为数字、字符串等基本数据类型属于包装类型会包装成对象

目录

Contents

- 深入对象
- 内置构造函数
- 综合案例

02

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]
```

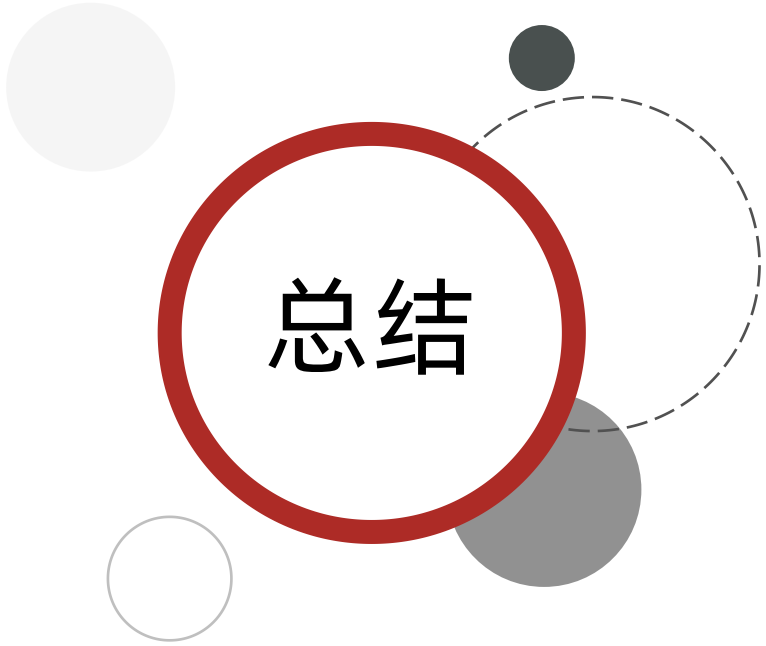
- **注意：** 返回的是一个数组

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}
```



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）
   - 返回的是数组
3. Object.values()方法的作用是什么？
   - 获取对象中所有属性值（值）
   - 返回的是数组

练习 请完成以下需求

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

请将size和color里面的值拼接为字符串之后，写到div标签里面，展示如下：

40cm*40cm/黑色

练习 请完成以下需求

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

请将size和color里面的值拼接为字符串之后，写到div标签里面，展示如下：

40cm*40cm/黑色

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
```

02

内置构造函数

- Object
- Array
- String
- Number

2.2 Array

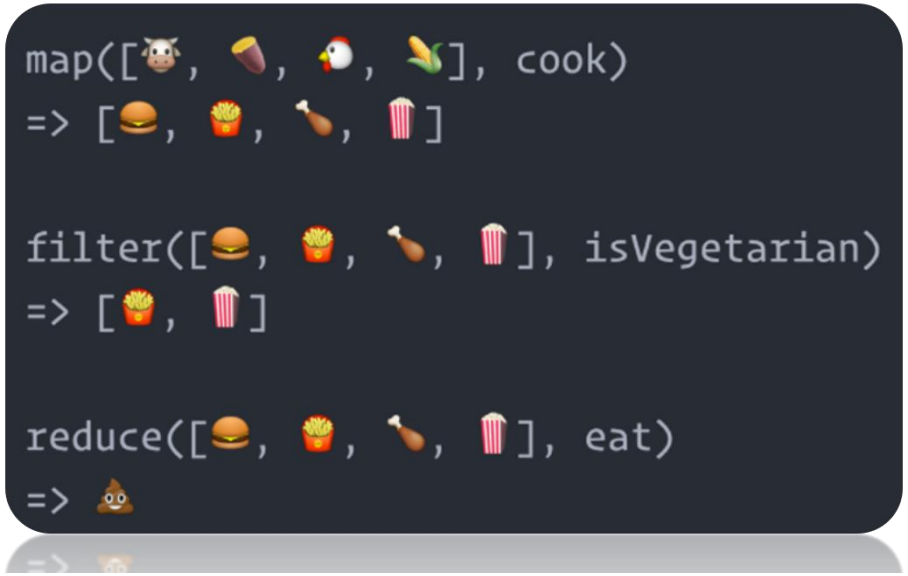
Array 是内置的构造函数，用于创建数组

```
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	不返回数组，经常用于查找遍历数组元素
filter	过滤数组	返回新数组，返回的是筛选满足条件的数组元素
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
reduce	累计器	返回累计处理的结果，经常用于求和等

2.2 Array

- 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]
```

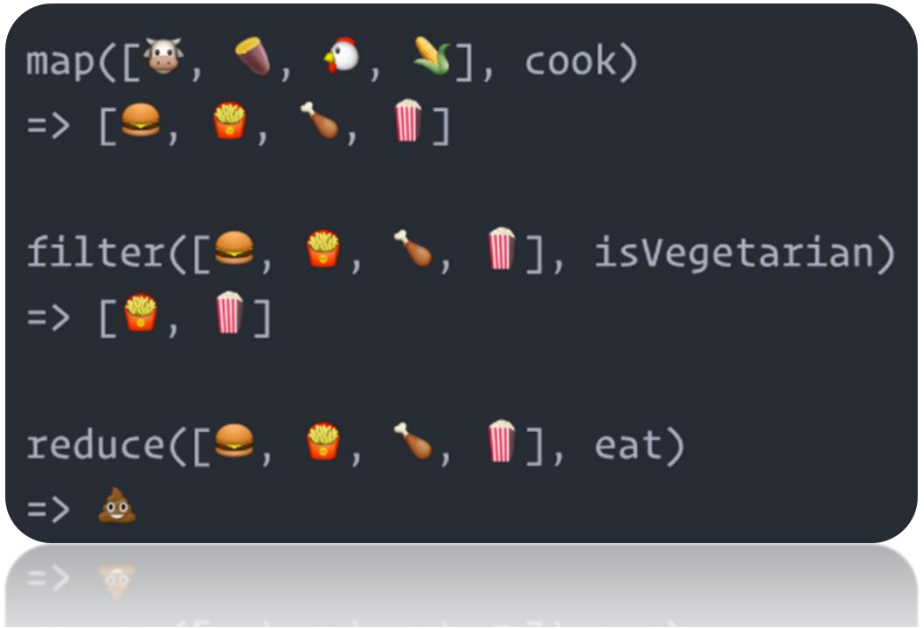
- 参数：
  1. 如果有起始值，则把初始值累加到里面

2.2 Array

- 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
filter	过滤数组	返回新数组，返回的是筛选满足条件的数组元素
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
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：

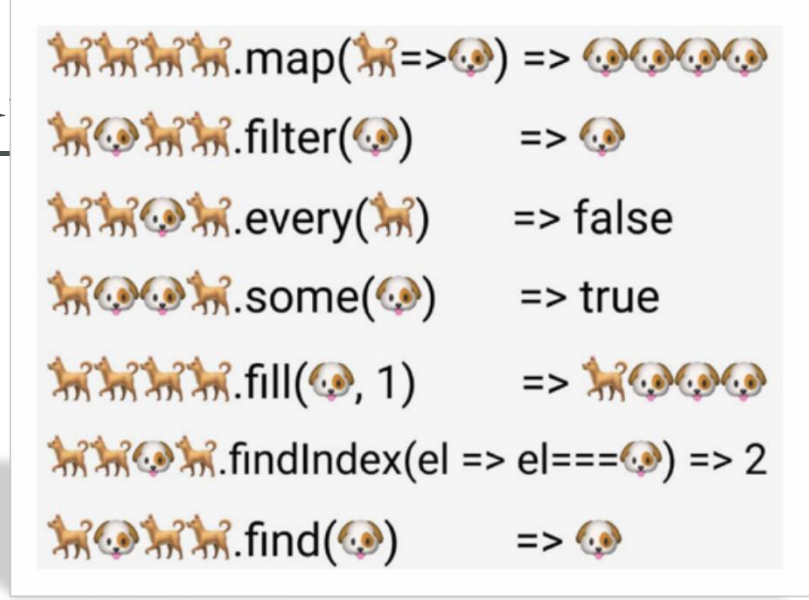
①：根据数据计算当月支出薪资

数据：

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

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，如果没有符合条件的则返回 undefined(重点)
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**都通过检测返回 true，否则返回 false(重点)
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**元素满足条件返回 true，否则返回 false
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

2.2 Array

2. 数组常见方法- 伪数组转换为真数组

常见伪数组

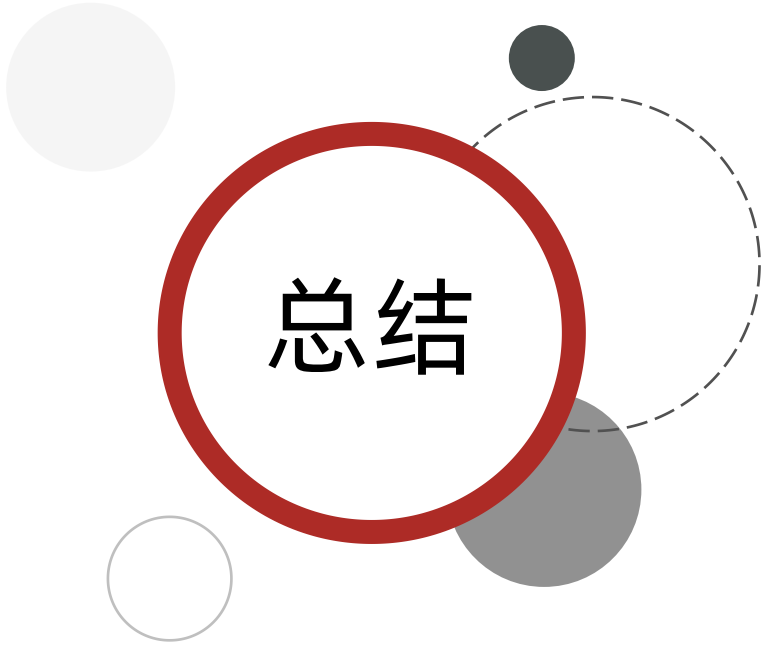
1. 函数里的arguments对象
2. querySelectorAll 等获得
3. 元素.children

注意：

querySelectorAll 里面可以使用 forEach方法

转换为真数组

- 静态方法 Array.from()
- 返回：新数组（真数组）
- 不修改原来的伪数组



1. 将伪数组转换为真数组我们学了哪个方法？
   - Array.from() 是个静态方法
2. 返回值是什么？影响原来的伪数组吗？
   - 返回真数组，不影响原来伪数组

案例

全选案例

需求①：点击全选，则下面小复选框自动全部勾选

需求②：小复选框全部勾选，则全选会自动勾选

全选	商品	商家	价格
☑	小米手机	小米	￥1999
☑	小米净水器	小米	￥4999
☑	小米电视	小米	￥5999

步骤

全选案例

需求①：点击全选，则下面小复选框自动全部勾选

实现：

- 点击全选，利用 forEach 让所有小复选框状态修正为全选状态

需求②：小复选框全部勾选，则全选会自动勾选

实现：

- 每次点击小复选框，利用every遍历所有小复选框，把返回值作为状态给全选按钮
- 注意转换为真数组才能使用every方法

全选	商品	商家	价格
☑	小米手机	小米	¥1999
☑	小米净水器	小米	¥4999
☑	小米电视	小米	¥5999

02

内置构造函数

- Object
- Array
- String
- Number

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 `trim()` 字符串的两端清除空格，返回一个新的字符串，而不修改原始字符串
7. 实例方法 `toUpperCase` 用于将字母转换成大写
8. 实例方法 `toLowerCase` 用于将就转换成小写
9. 实例方法 `indexOf` 检测是否包含某字符
10. 实例方法 `endsWith` 检测是否以某字符结尾
11. 实例方法 `replace` 用于替换字符串，支持正则匹配
12. 实例方法 `match` 用于查找字符串，支持正则匹配

练习

字符串翻转

请把'传智播客' 四个字翻转为 '客播智传'

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 用哪个方法？ split(',')

②：渲染页面： map + join 方法

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('')
```

02

内置构造函数

- Object
- Array
- String
- Number

2.4 Number（导师讲解）

Number 是内置的构造函数，用于创建数值

常用方法：

toFixed(保留位数长度) 设置保留小数位的长度

```
// 数字 toFixed 方法
const num = 12.345
console.log(num.toFixed(2))  // 12.35
console.log(num.toFixed(1))  // 12.3
const num1 = 12
console.log(num1.toFixed(2))  // 12.00
```

目录

Contents

- 深入对象
- 内置构造函数
- 综合案例

案例

购物车展示

需求：

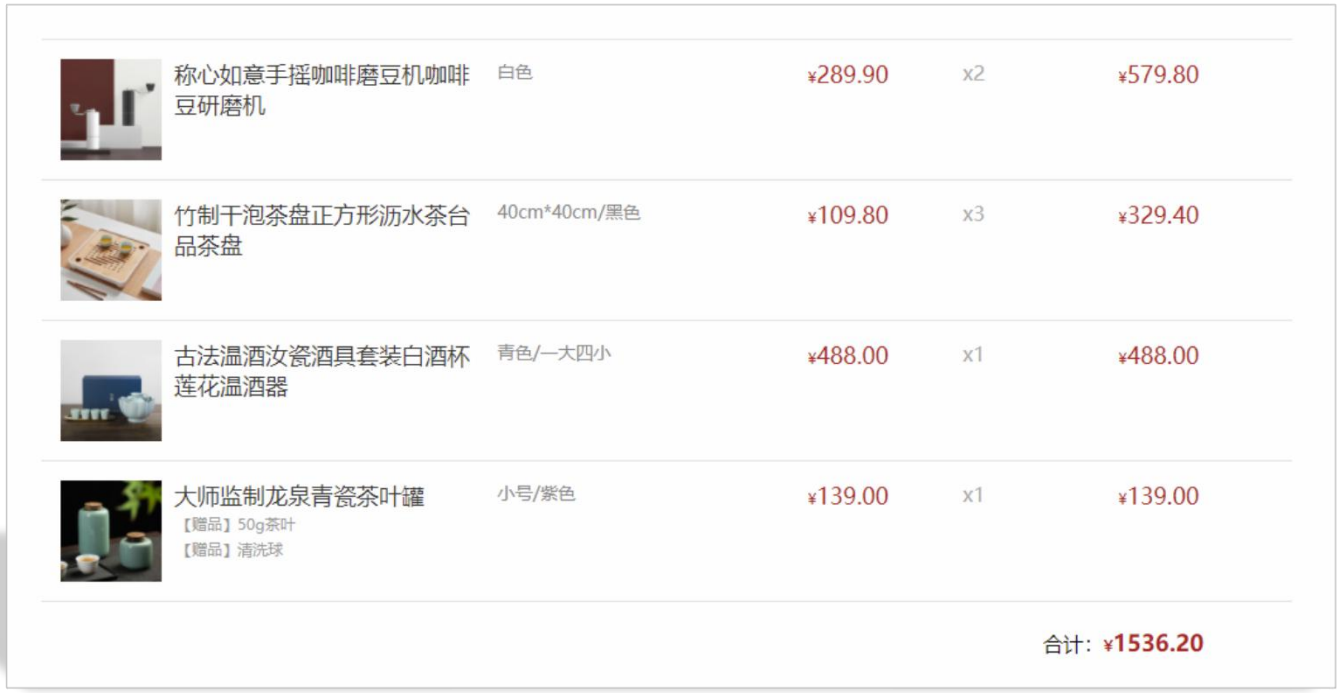
根据后台提供的数据，渲染购物车页面

	称心如意手摇咖啡磨豆机咖啡豆研磨机	白色	¥289.90	x2	¥579.80
	竹制干泡茶盘正方形沥水茶台品茶盘	40cm*40cm/黑色	¥109.80	x3	¥329.40
	古法温酒汝瓷酒具套装白酒杯莲花温酒器	青色/一大四小	¥488.00	x1	¥488.00
	大师监制龙泉青瓷茶叶罐 【赠品】50g茶叶 【赠品】清洗球	小号/紫色	¥139.00	x1	¥139.00

合计：¥1536.20

案例 **购物车展示**

分析业务模块：



- 渲染业务
- 数据处理业务
- 总价业务

步骤 **购物车展示**

渲染业务

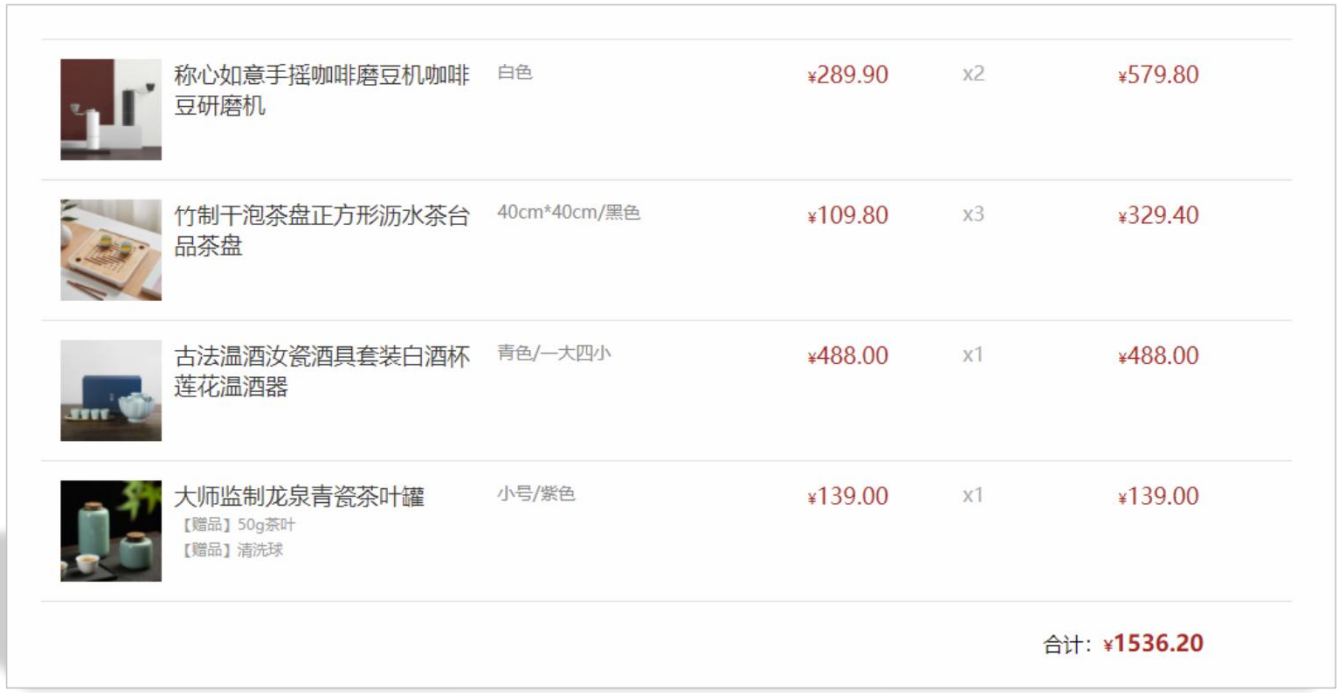
①：渲染业务

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)

案例 **购物车展示**

分析业务模块：



- 渲染业务
- 数据处理业务
- 总价业务

步骤 **购物车展示**

数据处理业务

②：数据处理业务

1. 处理 规格文字 模块

- 获取 每个对象里面的 spec，对象解构添加 spec
- 获得所有属性值是： Object.values() 返回的是数组
- 数组转换为字符串 join('/') 把字符串数据放入对应盒子里面即可


```
{
  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**

数据处理业务

②：数据处理业务

2. 处理 赠品 模块

- 获取 每个对象里面的 gift，对象解构添加 gift
- 把字符串转换为数组 split(',')，然后利用 map + join 生成对应标签， 放入对应盒子中
- 注意要判断是否有 gift 属性，没有的话不需要 map + join 生成标签


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

步骤 **购物车展示**

数据处理业务

②：数据处理业务

3. 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

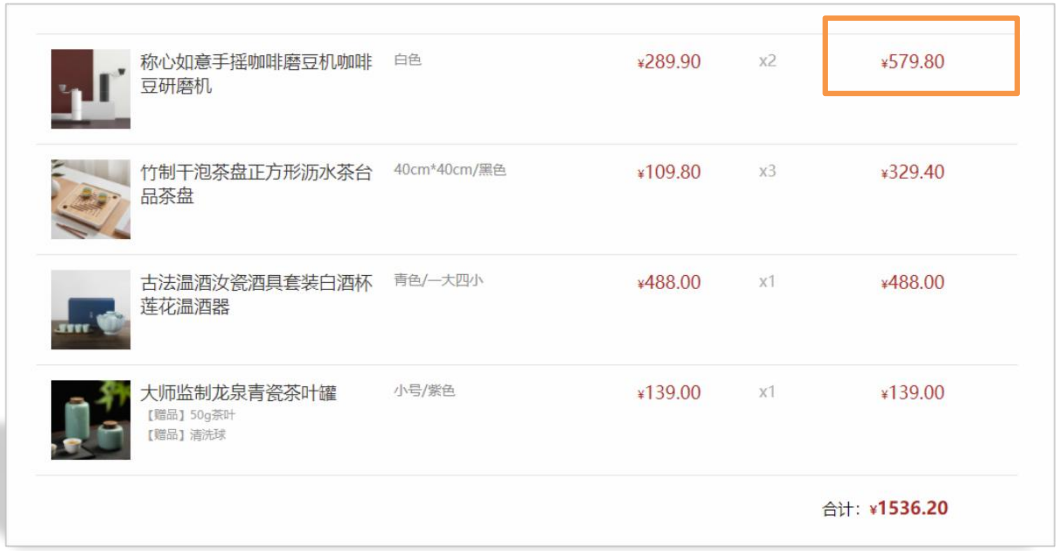
关于小数的计算精度问题:

0.1 + 0.2 = ?

解决方案： 我们经常转换为整数

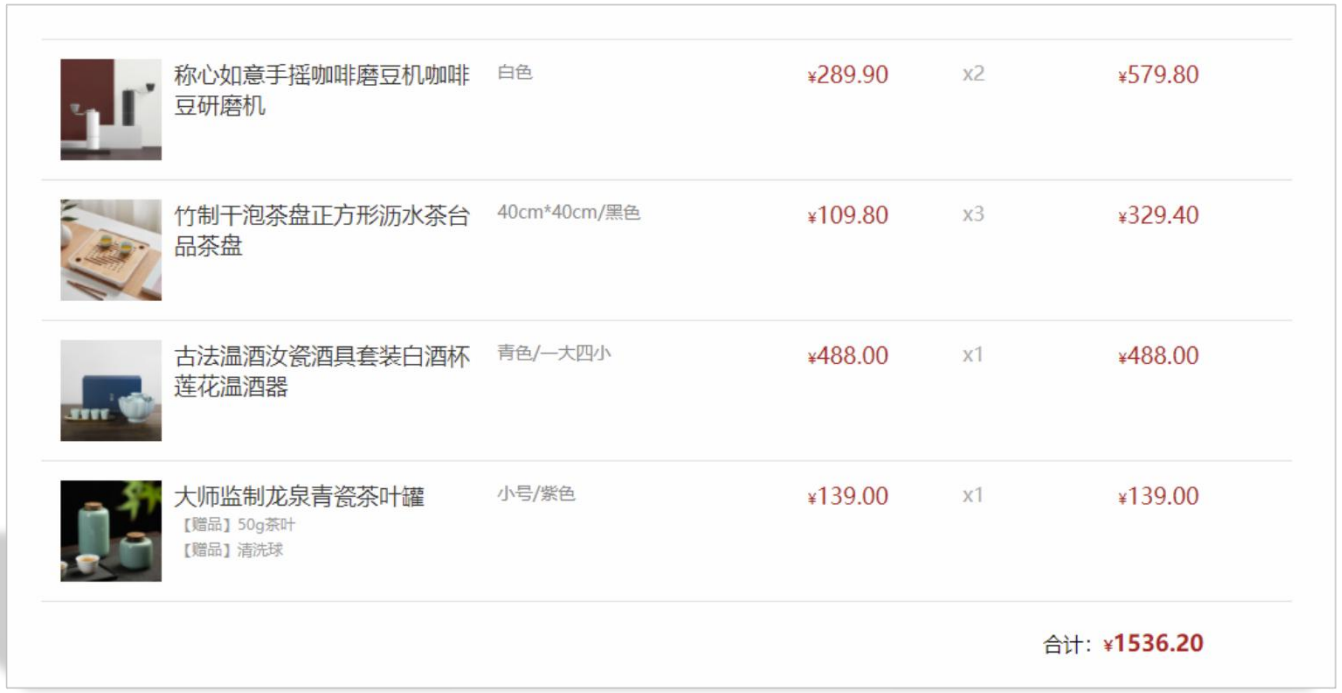
（0.1*10 + 0.2*10）/ 10 === 0.3

这里是给大家拓展思路和处理方案



案例 **购物车展示**

分析业务模块：



- 渲染业务
- 数据处理业务
- 总价业务

步骤 **购物车展示**

总价业务

③：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，总价也要保留2位小数
- 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

传智教育旗下高端IT教育品牌