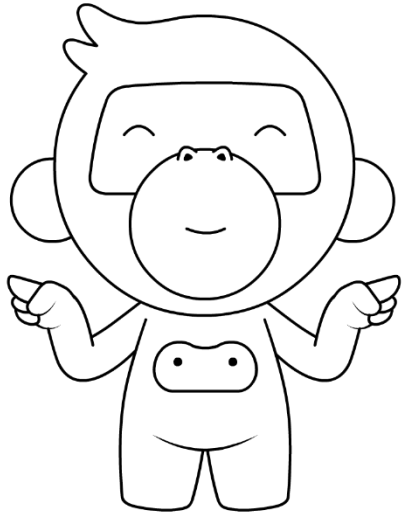
JavaScript 核心与进阶

Web 标准

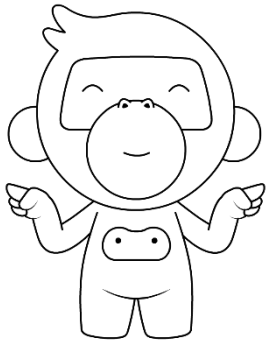


HTML 结构

CSS 样式

JavaScript 行为

Web 标准



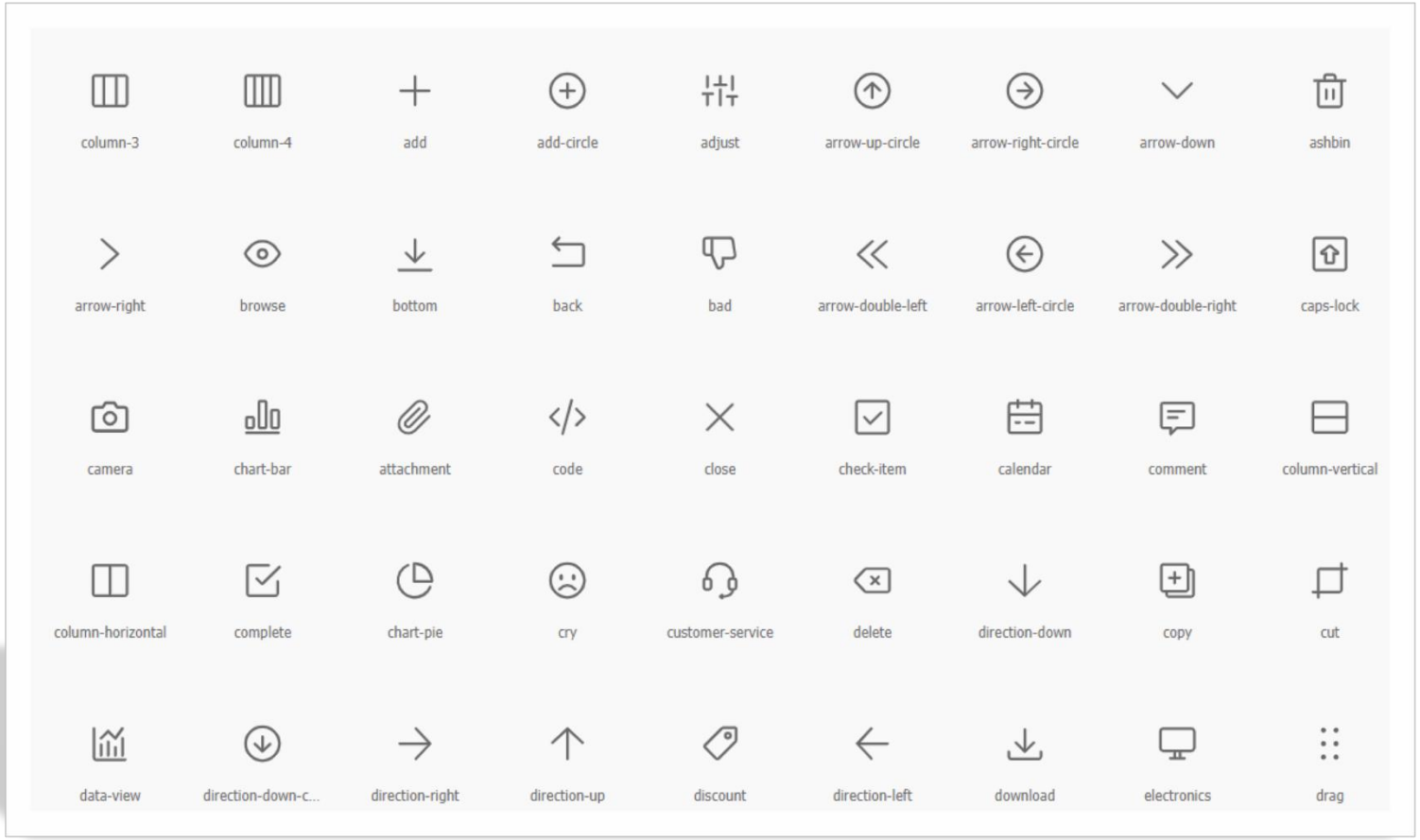
HTML 结构

CSS 样式

JavaScript 行为


好玩吗？重要吗？

JavaScript体验



JavaScript体验

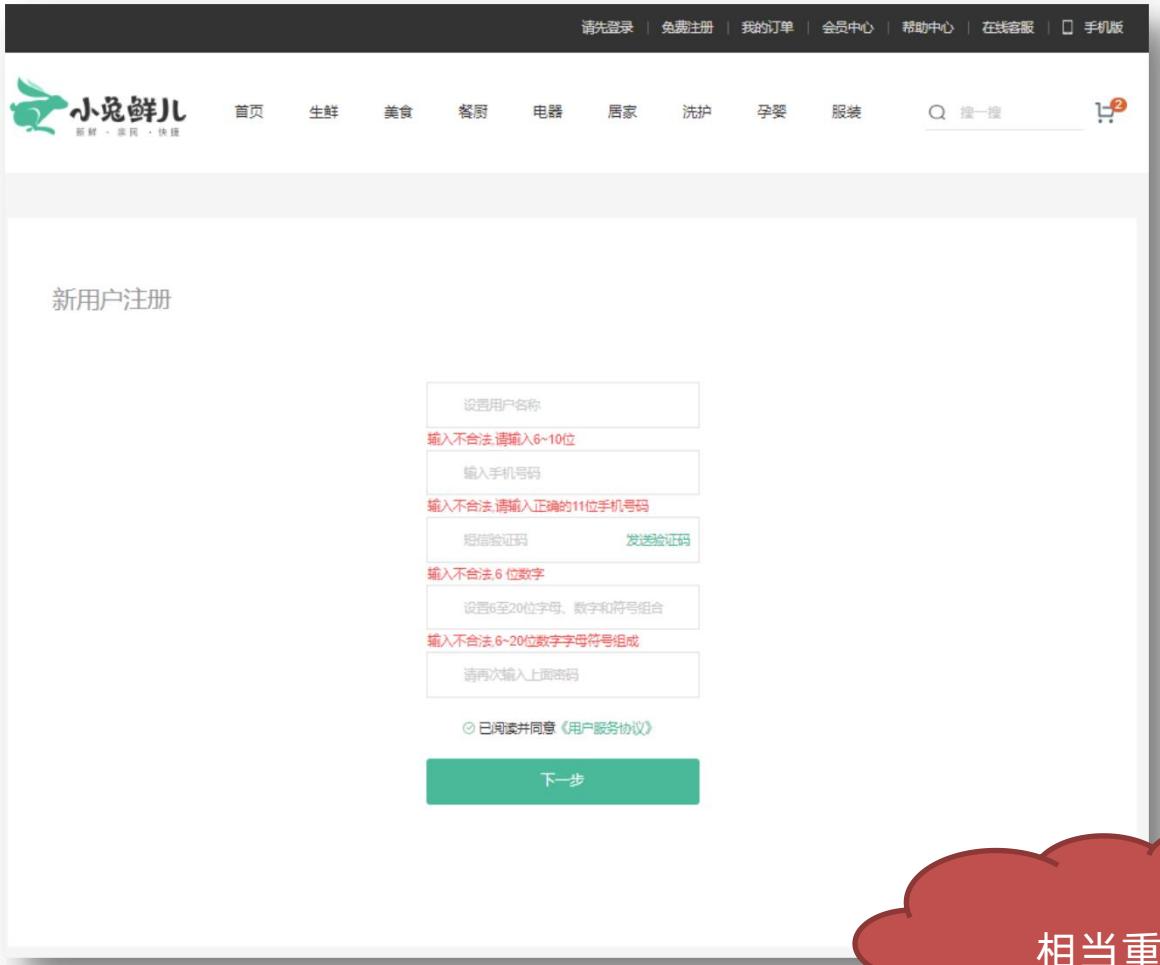
选择所有购物车小盒子

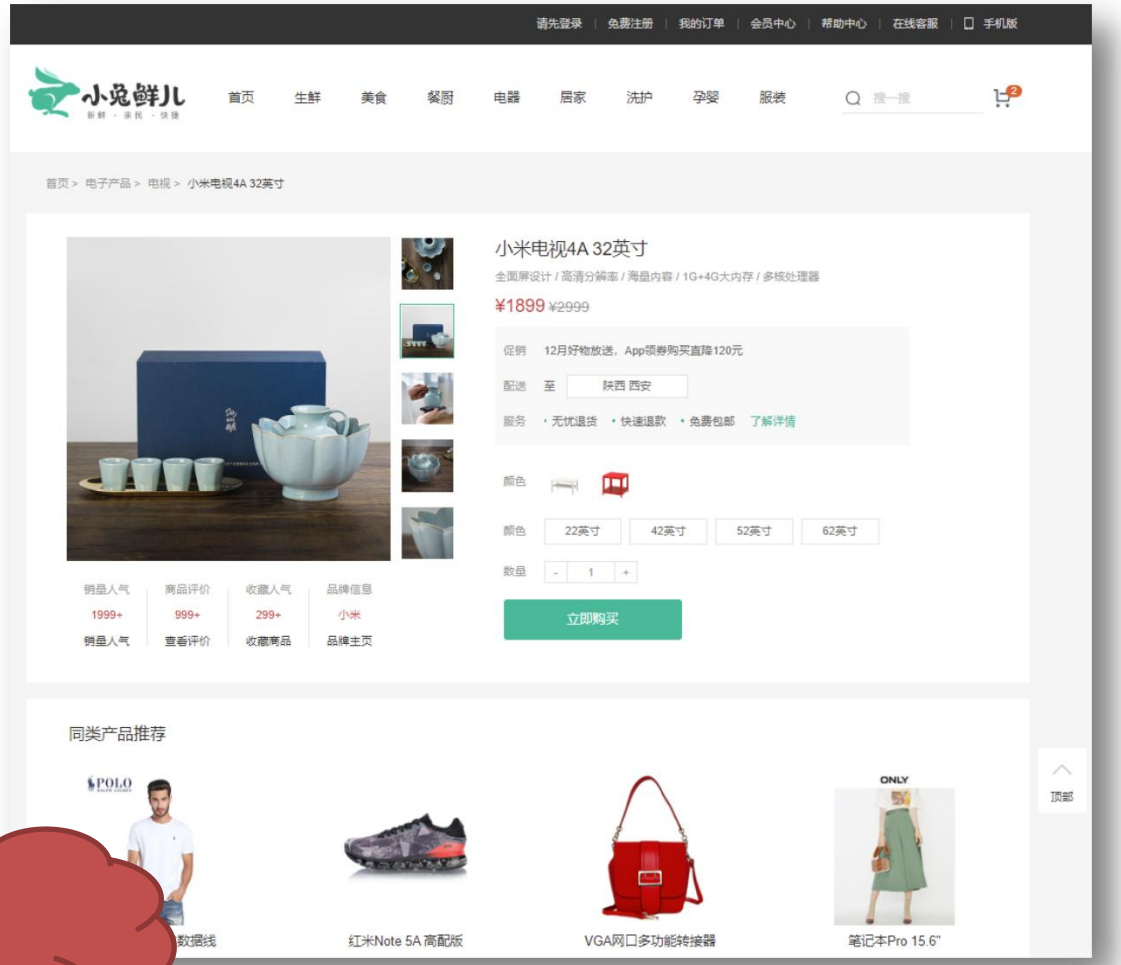
```
const items = document.querySelectorAll('.icon-cover span:nth-child(1)')

items.forEach(item => { item.click() })
```

利用js点击购物车小盒子


学完能干啥？





相当重要

注册、登录

网页特效


我们要学啥？

JavaScript 核心与进阶

JavaScript基础

5天

Web APIs

7天

JavaScript进阶

4天

基础：
基本写法，知道怎么控制计算机去执行代码....

应用：
操作页面元素做出各种好玩的效果....

进阶：
高级语法、高频面试题、底层原理....

JavaScript 基础第一天

变量、数据类型、运算符

描述	单词	效果
输入输出语句	alert('页面弹出警示框')	输出语句，页面弹出警示框
	document.write('页面打印输出')	输出语句，页面文档打印输出
	console.log('控制台打印输出')	输出语句，控制台输出
	prompt('请输入姓名:')	输入语句，输入对话框
声明变量	let num = 10	声明变量
声明常量	const pi = 3.14	声明常量
基本数据类型	number	数字类型
	string	字符串类型
	boolean	布尔类型
	undefined	未定义类型
	null	空类型
运算符	算术运算符、比较运算符、逻辑运算符、一元运算符、赋值运算符	

目录

Contents

JavaScript 介绍

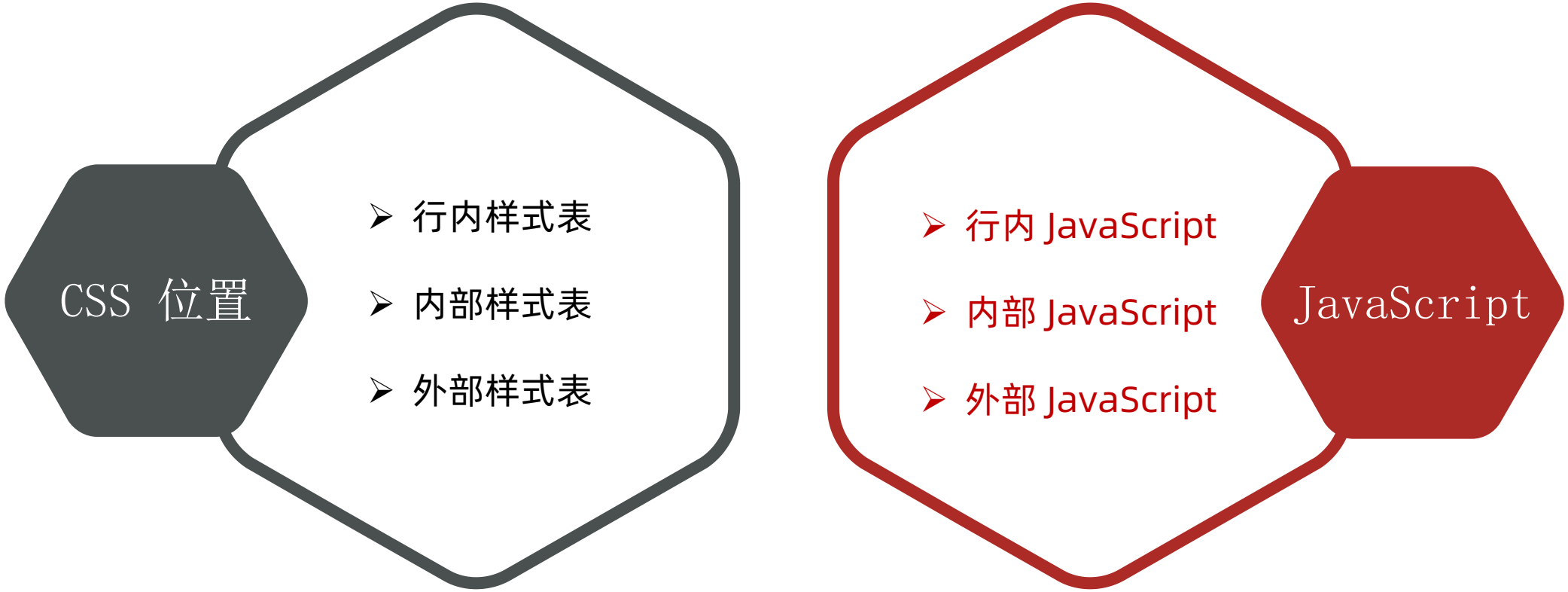
- JavaScript 是什么？怎么写？
- JavaScript 输入和输出语句

1.1 JavaScript 是什么

是一种运行在客户端（浏览器）的编程语言，可以用来创建动态更新的内容，控制多媒体，制作图像动画等交互效果

1.2 JavaScript 书写位置

- 目标：知道如何向页面添加 JavaScript



1.2 JavaScript 书写位置

1. 内部 JavaScript

直接写在html文件里，用script标签包住

规范：script标签写在</body>上面

拓展： alert('你好，JavaScript') 页面弹出警告对话框

```
<script>
  alert('你好,JavaScript~')
</script>
```

注意事项

我们将〈script〉放在HTML文件的底部附近的原因是浏览器会按照代码在文件中的顺序加载 HTML。
如果先加载的 JavaScript 期望修改其下方的 HTML，那么它可能由于 HTML 尚未被加载而失效。
因此，将 JavaScript 代码放在 HTML页面的底部附近通常是最好的策略。

1.2 JavaScript 书写位置

2. 外部 JavaScript

代码写在以.js结尾的文件里

语法：通过script标签，引入到html页面中。

```
<body>
    <!-- 通过src引入外部js文件 -->
    <script src="my.js"></script>
</body>
```

注意事项

1. script标签中间无需写代码，否则会被忽略！
2. 外部JavaScript会使代码更加有序，更易于复用，且没有了脚本的混合，HTML 也会更加易读，因此这是个好的习惯。

1.2 JavaScript 书写位置

3. 行内 JavaScript

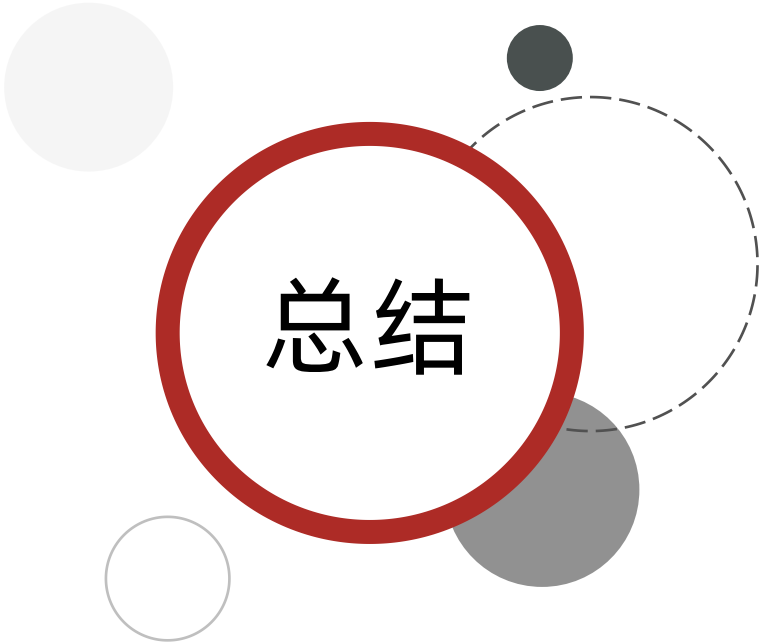
代码写在标签内部

语法：直接标签内部写js代码

```
<body>
    <button onclick="alert('逗你玩~~~')">点击我月薪过万</button>
</body>
```

注意事项

此处作为了解即可，但是后面vue框架会用这种模式



1. JavaScript是什么？
   - JavaScript 是一种运行在客户端（浏览器）的编程语言
2. JavaScript代码可以写到哪三种位置？
   - 内部
   - 外部
   - 行内
3. 注意事项：
   - 书写的位置尽量写到文档末尾 </body>标签前面
   - 外部 js 标签中间不要写代码，否则会被忽略

```
<body>
    <!-- 通过src引入外部js文件 -->
    <script src="my.js"></script>
</body>
```

练习

页面弹框课堂练习

需求：

①：请用 JavaScript 内部 书写方式，页面弹出： 努力，奋斗

②：请用 JavaScript 外部 书写方式，页面弹出： 月薪过万

时间：5分钟

1.2 JavaScript 怎么写

目标：了解JavaScript注释写法以及结束符通俗写法

- 注释
- 结束符

```
<script>
  // alert弹出警示框
  alert('你好,js')
</script>
```

1.2 JavaScript 怎么写

- 单行注释
  - **符号：**//
  - **作用：**//右边这一行的代码会被忽略
  - **快捷键：**ctrl + /

```
<script>
  // 这种是单行注释的语法
  // 一次只能注释一行
  // 可以重复注释
</script>
```

- 块注释
  - **符号：**/* */
  - **作用：**在/* 和 */ 之间的所有内容都会被忽略
  - **快捷键：**shift + alt + a

```
<script>
  /* 这种的是多行注释的语法 */
  /*
     更常见的多行注释是这种写法
     在些可以任意换行
     多少行都可以
  */
</script>
```

1.2 JavaScript 怎么写

- 结束符
  - **作用：** 使用英文的 ；代表语句结束
  - **实际情况：** 实际开发中，可写可不写，浏览器(JavaScript 引擎) 可以自动推断语句的结束位置
  - **现状：** 在实际开发中，越来越多的人主张，书写 JavaScript 代码时省略结束符
  - **约定：**为了风格统一，结束符要么每句都写，要么每句都不写（按照团队要求.）

```
<script>
  alert(1);
  alert(2);
</script>
```

```
<script>
  alert(1)
  alert(2)
</script>
```

1.2 JavaScript 怎么写

- 结束符

JavaScript 语句后应该加分号么?

尤雨溪
前端开发话题下的优秀答主

+ 关注

939 人赞同了该回答

没有应该不应该，只有你自己喜欢不喜欢。JavaScript 语法长得 C-like 不代表它本质上和 C 是一类语言，所有直觉性的 “当然应该加分号” 都是保守的、未经深入思考的草率结论。后来新设计的语言里可选分号的多得去了，光是 “可以加分号但是大家都不加” 的语言就有：Go, Scala, Ruby, Python, Swift, Groovy...

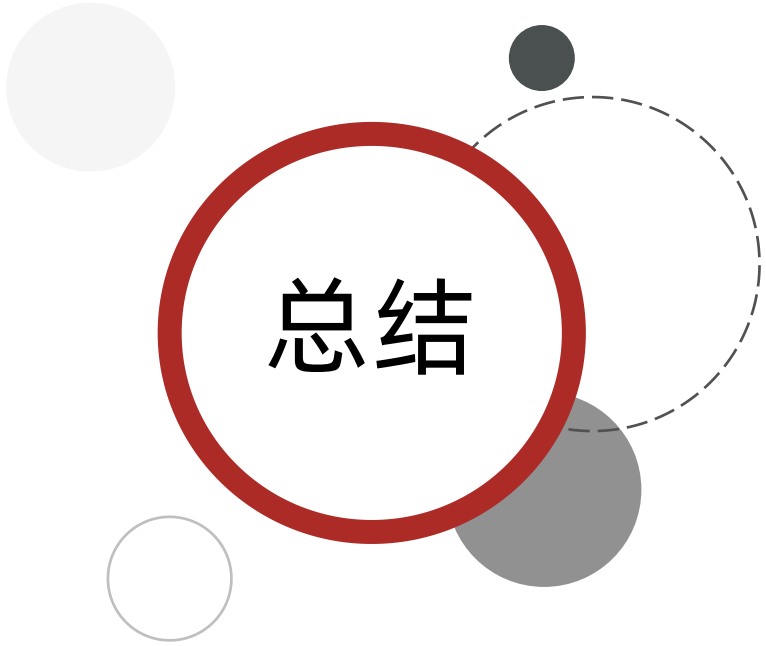
至于说 “很难总结什么时候加不加”，其实真的很简单。真正会导致上下行解析出问题的 token 有 5 个：括号，方括号，正则开头的斜杠，加号，减号。我还从没见过实际代码中用正则、加号、减号作为行首的情况，所以总结下来就是一句话：一行开头是括号或者方括号的时候加上分号就可以了，其他时候全部不需要。其实即使是这两种情况，在实际代码中也颇为少见。

另外，restricted production 这个东西（也就是导致 return 后面换行会自动插入分号的机制），不管你加不加分号你都是得搞懂了才能不被坑的，和加不加分号没有什么关系。

更多细节，可以看我曾经给过的一个 talk：

Hacking Semicolons by Evan You

最后，上点代码好了，Vue.js 的代码全部不带分号：



1. JavaScript 注释有哪两种方式？
   - 单行注释 //
   - 多行注释 /* */
2. JavaScript 结束符注意点
   - 结束符是？
     - ✓ 分号；
   - 结束符可以省略吗？
     - ✓ Yes
     - ✓ 但为了风格统一，结束符要么每句都写，要么每句都不写（团队约定）

JavaScript 介绍

- JavaScript 是什么？怎么写？
- 输入和输出语句

1.3 JavaScript 输入输出语句（重点）

目标：能写出常见 JavaScript 输入输出语句

交互： 用户输入数据 → 数据处理 → 输出结果展示给用户

学习路线：

- 输出语句
- 输入语句

1.3 JavaScript 输入输出语句

1. 输出语句：

语句1：

```
alert('页面警示框输出的内容')
```

作用：页面弹出警告框

语句2：

```
document.write('向页面文档输出的内容')
```

作用：在body标签内输出内容

提示：如果输出的内容写标签，也会被解析成网页元素

语句3：

```
console.log('控制台输出内容')
```

作用：控制台输出语法，程序员调试使用

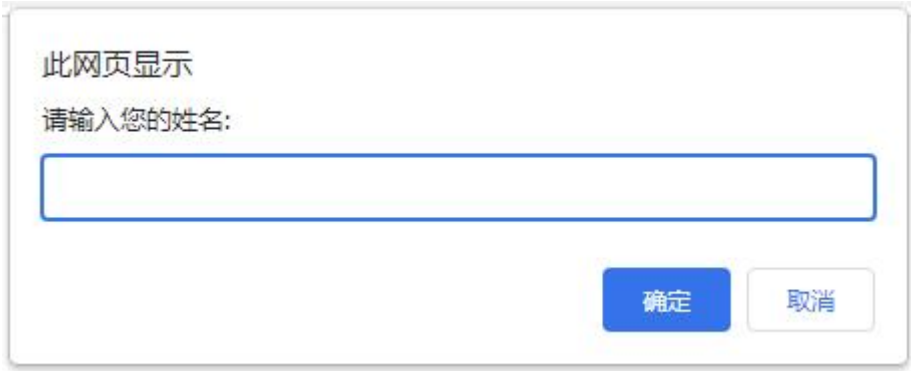
1.3 JavaScript 输入输出语句

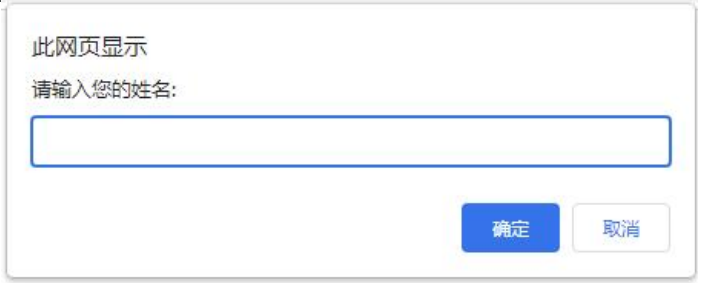
2. 输入语句：

- 语法：

```
prompt('请输入您的姓名:')
```

- prompt　提示的意思
- 作用：显示一个对话框，对话框中包含一条文字信息，用来提示用户输入文字
- 展示：





总结

1.我们讲了哪些 JavaScript 输出语句？

- document.write(' 我是div标签 ')　在body中输出内容，能解析标签
- alert('请勾选同意协议')　页面弹出警告框
- console.log('控制台打印')　控制台输出，主要给程序员调试使用

2.我们讲了哪个 JavaScript 输入语句？

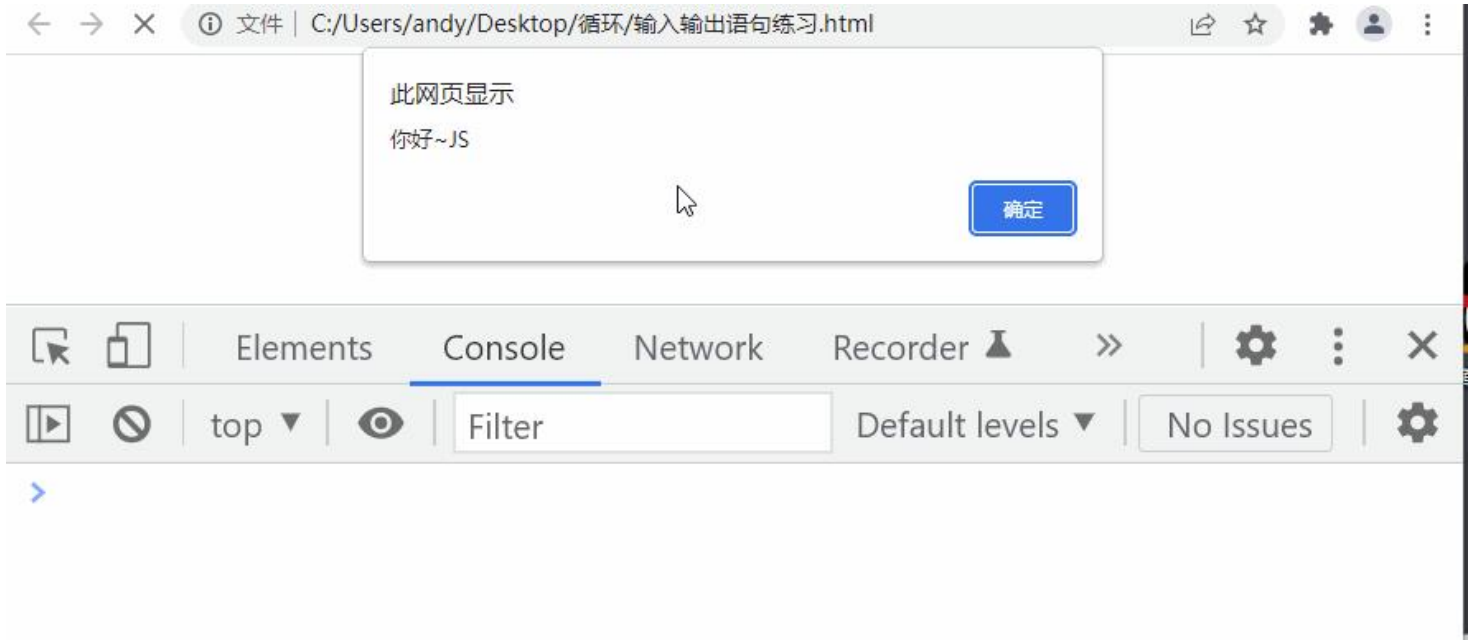
- prompt('请输入姓名:')　页面弹出对话框，可以提示用户输入内容

练习

输入和输入练习：时间 5分钟（同学自行练习）

需求：

1. 浏览器中弹出警示框： 你好~ JS　　alert()
2. 页面中打印输出： JavaScript 我来了！　　document.write()
3. 页面控制台输出： 它~会魔法吧~　　console.log()



目录

Contents

02

变量

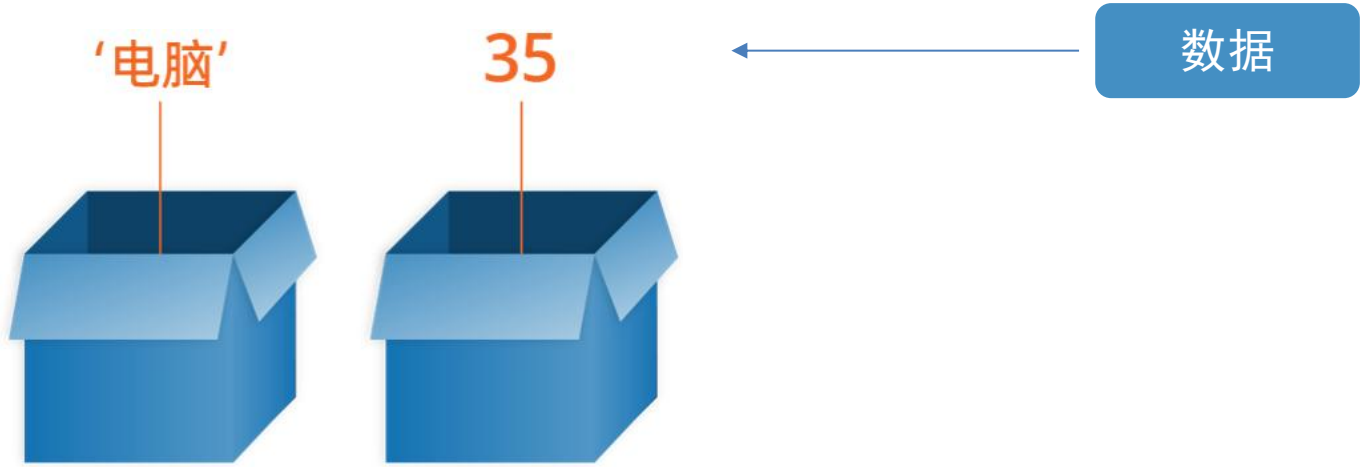
- 变量是什么&基本使用
- 变量的本质
- 变量命名规则与规范

2.1 变量是什么？

目标：理解变量是计算机存储数据的“容器”

1. **变量：**

- **是什么：**变量是计算机中用来**存储数据**的“容器”（盒子）
- **作用：**记录计算机中数据的不同状态



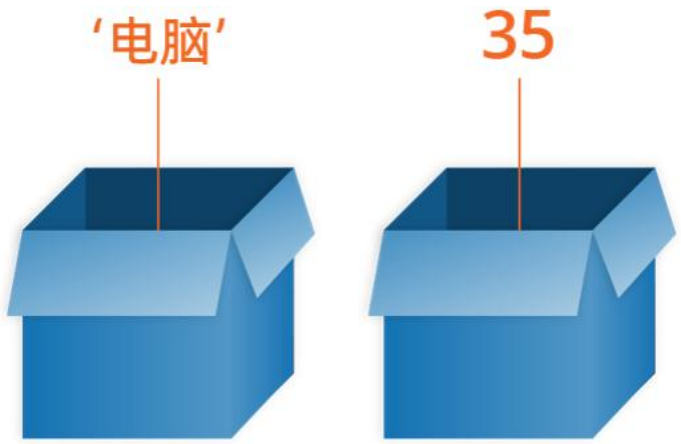
- 注意：变量不是数据本身，它们仅仅是一个用于存储数值的容器。可以理解为是一个个用来装东西的纸箱子。

2.2 变量的基本使用

1. 变量的声明（创建变量）
2. 变量的赋值（把数据存储到变量里面）

注意：

1. 数字直接存储
2. 字符用单引号引起来，表示一段信息



2.2 变量的基本使用

1. 声明变量：

要想使用变量，首先需要创建变量（也称为声明变量或者定义变量）

语法：

```
let 变量名
```

- 声明变量有两部分构成：声明关键字、变量名（标识）
- let 即关键字（let：允许、许可、让、要），所谓关键字是系统提供的专门用来声明（定义）变量的词语

举例：

```
let age
```

- 我们声明了一个age变量
- age 即变量的名称，也叫 标识符

2.2 变量的基本使用

2. 变量赋值：

定义了一个变量后，你就能够初始化它（赋值）。在变量名之后跟上一个“=”，然后是数值。

注意：是通过变量名来获得变量里面的数据

2.2 变量的基本使用

2. 变量赋值：

简单点，也可以声明变量的时候直接完成赋值操作，这种操作也称为 变量初始化。

```
<script>
    // 声明了一个age变量，同时里面存放了 18 这个数据
    let age = 18
</script>
```

总结

1. 变量是什么？
   - 计算机中用来存储数据的“容器”
2. 使用变量分为哪两步？
   - let 声明变量
   - = 来赋值数据
3. 开发中我们经常声明的同时可以直接赋值怎么写？

```
<script>
    // 声明了一个age变量，同时里面存放了 18 这个数据
    let age = 18
</script>
```

练习

- **课堂变量练习（同学自行练习）**

需求：

1. 声明一个变量，用于存放用户购买的商品 数量（ num ）为 20 件
2. 声明一个变量，用于存放用户的 姓名（ uname ）为 ’张三’
3. 依次**控制台**打印输出两个变量

2.2 变量的基本使用

目标：掌握变量的更新以及了解同时声明多个变量的简写方式

3. 更新变量（更换变量里面的值）

4. 同时声明多个变量　（同时声明多个变量的简写方式）

```
let sex = '男'
let uname = 'pink老师'
let age = 18
```

2.2 变量的基本使用

目标：掌握变量的更新以及了解同时声明多个变量的写法

3. 更新变量：

变量赋值后，还可以通过简单地给它一个不同的值来更新它。

```
// 声明了一个age变量，同时里面存放了 18 这个数据
let age = 18
// 变量里面的数据发生变化更改为 19
age = 19
// 页面输出的结果为 19
document.write(age)
```

```
// 声明了一个age变量，
let age = 18
// 这里不允许多次声
let age = 19
// 输出会报错
document.write(age)
```

注意： let 不允对同一个变量多次声明

2.2 变量的基本使用

4. 声明多个变量：

语法：多个变量中间用逗号隔开。

```
let age = 18, uname = 'pink'
```

说明：看上去代码长度更短，但并**不推荐**这样。为了更好的可读性，一般情况下我们一行只声明一个变量。

```
//多行变量声明有点长，但更容易阅读
let age = 18
let uname = 'pink'
```

总结

1. 变量赋值之后如何更新新值？
   - 直接给它一个不同的值来更新它
2. 如果出现以下错误信息原因是啥？

   - 多次声明了age变量，let不允许多次声明同一个变量
3. 我们提倡同时声明多个变量的简写方式吗？

```
let uname = 'pink老师', sex = '男'
```

   - 不提倡， 可读性不好

```
//多行变量声明有点长，但更容易阅读
let age = 18
let uname = 'pink'
```

变量拓展-let和var的区别（导师讲解）

let 和 var 区别：

在较旧的JavaScript，使用关键字 var 来声明变量 ，而不是 let。

var 现在开发中一般不再使用它，只是我们可能再老版程序中看到它。

let 为了解决 var 的一些问题。

var 声明：

- 可以先使用 在声明（不合理）
- var 声明过的变量可以重复声明(不合理)
- 比如变量提升、全局变量、没有块级作用域等等

结论：

var 就是个bug，别迷恋它了，以后声明变量我们统一使用 let

案例

1. 变量案例-输出姓名

目的： 练习变量保存数据以及JavaScript处理数据的流程

需求：

①：浏览器中弹出对话框： 请输入姓名

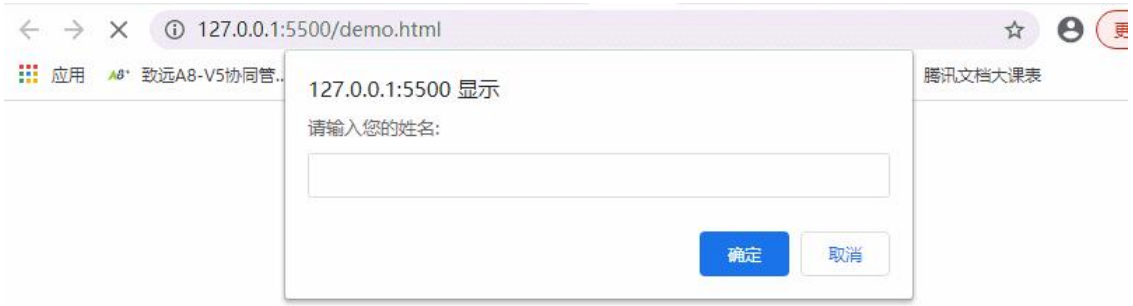
②：页面中输出：刚才输入的姓名

分析：

①：输入：用户输入框： prompt()

②：保存数据 （变量）

③：输出： 页面输出数据 document.write()



案例

2.变量案例- 交换变量的值

目的： 练习变量存储和常见面试题

需求：

①：有2个变量： apple 里面放的是 ‘苹果汁’， orange 里面放的是 ‘橙子汁’

②：经过处理，最后 apple 里面放的是 ‘橙子汁’， orange 里面放的是 ‘苹果汁’

案例

2.变量案例- 交换变量的值

分析：

思路：借助于一个空杯子

案例

2.变量案例- 交换变量的值

分析：

思路：借助于一个空杯子

空杯子

案例

2.变量案例- 交换变量的值

分析：

思路：借助于一个空杯子

案例

2.变量案例- 交换变量的值

分析：

思路：借助于一个空杯子

案例

2.变量案例- 交换变量的值

转换为代码分析：

1. 核心思路：使用一个 空变量 用来做临时中间存储

步骤：

1. 声明一个临时变量 temp
2. 把apple的 苹果汁 倒入 temp （赋值）

案例

2.变量案例- 交换变量的值

转换为代码分析：

1. 核心思路：使用一个 空变量 用来做临时中间存储

步骤：

1. 声明一个临时变量 temp
2. 把apple的 苹果汁 倒入 temp （赋值）

案例

2.变量案例- 交换变量的值

转换为代码分析：

1. 核心思路：使用一个 空变量 用来做临时中间存储

步骤：

1. 声明一个临时变量 temp
2. 把apple的 苹果汁 倒入 temp （赋值）
3. 把orange橙子汁倒入 apple 杯子里面

案例

2.变量案例- 交换变量的值

转换为代码分析：

1. 核心思路：使用一个 空变量 用来做临时中间存储

步骤：

1. 声明一个临时变量 temp
2. 把apple的 苹果汁 倒入 temp （赋值）
3. 把orange橙子汁倒入 apple 杯子里面

案例

2.变量案例- 交换变量的值

转换为代码分析：

1. 核心思路：使用一个 空变量 用来做临时中间存储

步骤：

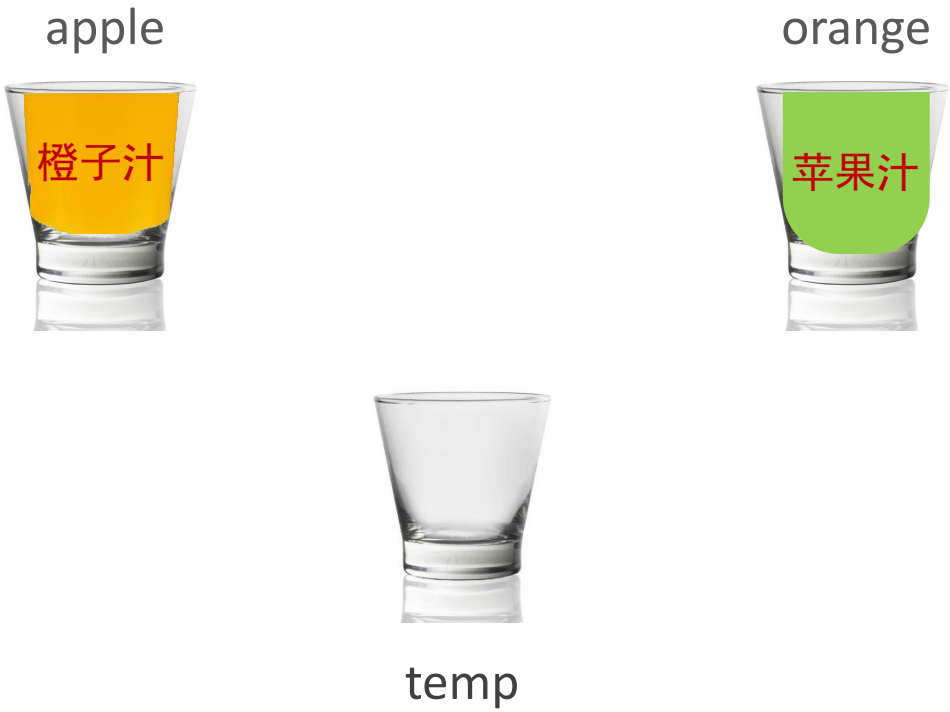
1. 声明一个临时变量 temp
2. 把apple的 苹果汁 倒入 temp （赋值）
3. 把orange橙子汁倒入 apple 杯子里面
4. 把 temp里面的苹果汁倒入 orange杯子

案例

2.变量案例- 交换变量的值

转换为代码分析：

1. 核心思路：使用一个 空变量 用来做临时中间存储



步骤：

1. 声明一个临时变量 temp
2. 把apple的 苹果汁 倒入 temp (赋值)
3. 把orange橙子汁倒入 apple 杯子里面
4. 把 temp里面的苹果汁倒入 orange杯子

02

变量

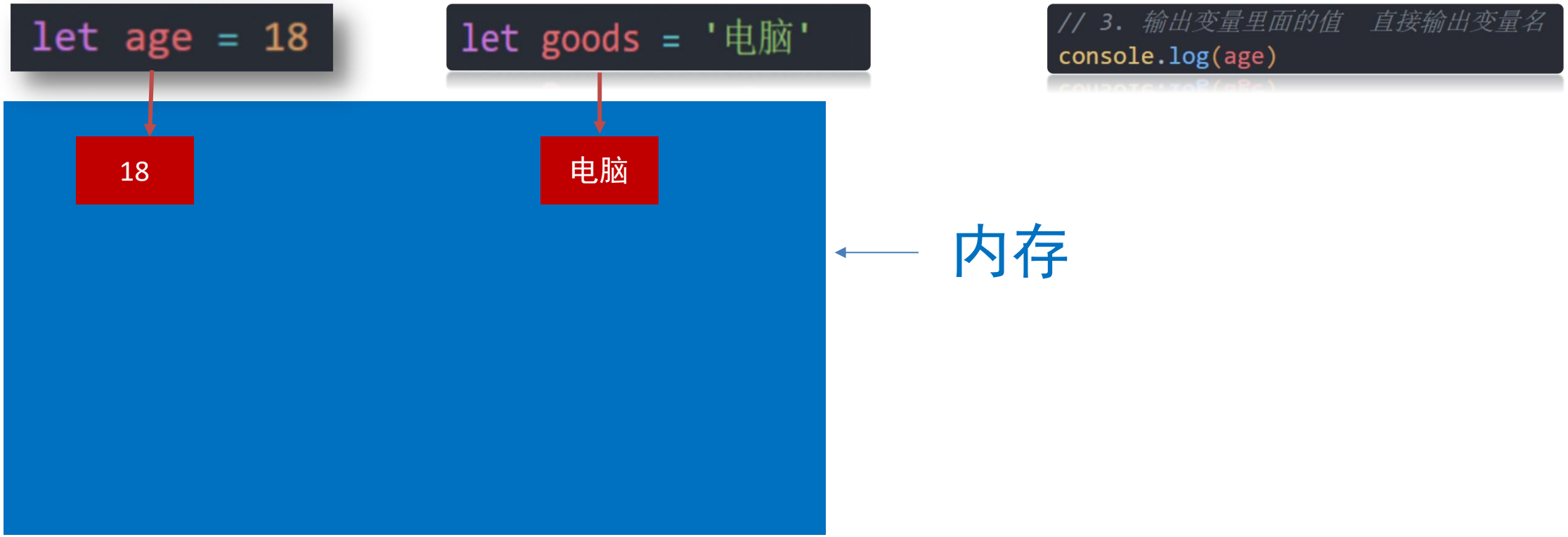
- 变量是什么&基本使用
- 变量的本质
- 变量命名规则与规范

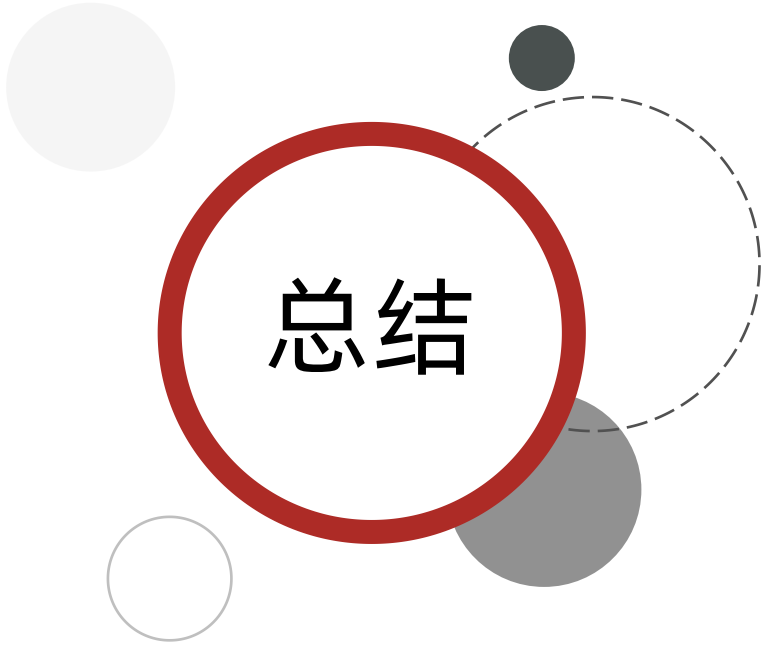
2.3 变量的本质

目标：能够说出变量的本质是什么

内存：计算机中存储数据的地方，相当于一个空间

变量本质：是程序在内存中申请的一块用来存放数据的小空间





1. 变量的本质是什么？
   - 是程序在内存中申请的一块用来存放数据的小空间
2. 变量很多时内存中会开辟很多小空间，如何找到对应的变量？
   - 通过变量名
   - 通过变量名可以到内存中找到对应的空间，就可以得到里面的数据

02

变量

- 变量是什么&基本使用
- 变量的本质
- 变量命名规则与规范

2.4 变量命名规则与规范

目标：能写出专业的变量名

规则：必须遵守，不遵守报错（法律层面）

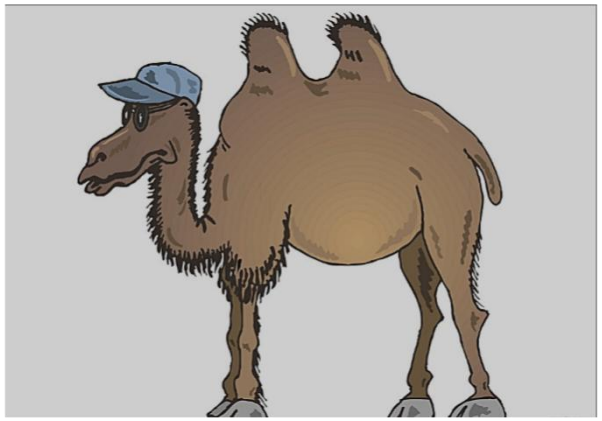
规范：建议，不遵守不会报错，但不符合业内通识 （道德层面）

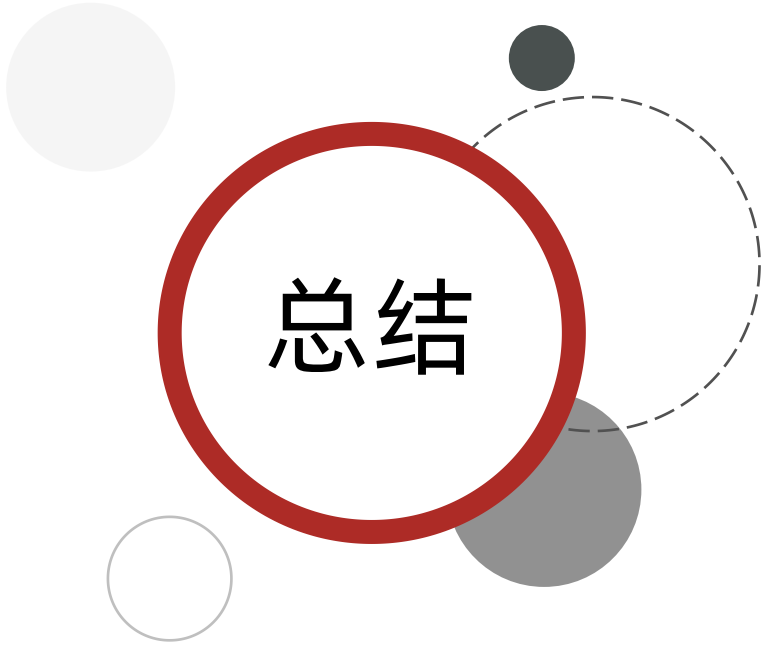
1. 规则：

- 不能用关键字
  - ✓ 关键字：有特殊含义的字符，JavaScript 内置的一些英语词汇。例如：let、var、if、for等
- 只能用下划线、字母、数字、$组成，且数字不能开头
- 字母严格区分大小写，如 Age 和 age 是不同的变量

2. 规范：

- 起名要有意义
- 遵守小驼峰命名法
  - ✓ 第一个单词首字母小写，后面每个单词首字母大写。例：userName





1. **规则：**（法律层面）
   - 不能用关键字
   - 只能用下划线、字母、数字、$组成，且数字不能开头
   - 字母严格区分大小写，如 Age 和 age 是不同的变量
2. **规范：**（道德层面）
   - 起名要有意义
   - 遵循小驼峰命名法

2.4 变量命名规则与规范（导师提问同学）

以下哪些是合法的变量名?

变量名	是否报错	是否符合规范
21age		
_age		
user-name		
username		
userName		
let		
na@me		
$age		

2.4 变量命名规则与规范

以下哪些是合法的变量名?

变量名	是否报错	是否符合规范
21age	报错	
_age	不报错	符合规范
user-name	报错	
username	不报错	不符合规范
userName	不报错	符合规范
let	报错	
na@me	报错	
$age	不报错	符合规范

练习

变量练习-输出用户信息（同学自行练习）

需求：让用户输入自己的名字、年龄、性别，再输出到网页

分析：

①：弹出 输入 框（prompt）： 请输入您的姓名 (uname)： 用变量保存起来。

②：弹出输入框（prompt）： 请输入您的年龄 (age)： 用变量保存起来。

③：弹出输入框（prompt）： 请输入您的性别(sex)： 用变量保存起来。

④：页面分别 输出 (document.write) 刚才的 3 个变量。

目录

Contents

常量

3. 常量

- **是什么：**也是一个容器，用于保存数据的
- **和变量的区别：** 常量里面保存的值是不允许改变的
- **使用场景：**当某个值永远**不会改变**的时候，我们可以使用常量来保存，目的为了程序的安全
- **常量使用：**

```
const pi = 3.14 // 声明常量
console.log(pi) // 输出常量里面的值
```

- const 单词 常量 的意思

3. 常量的基本使用

- **注意：**

  1. 常量不允许重新赋值，可以理解为是只读的
  2. 声明的时候必须赋值（初始化）

总结

1. 常量的值可以修改吗？ 可以不初始化吗？
   - 常量不允许重新赋值，可以理解为是只读的
   - 声明的时候必须赋值（初始化）
2. 如果出现以下错误信息会是什么问题引起的？

```
▶Uncaught TypeError:     11-常量.html:18
Assignment to constant variable.
    at 11-常量.html:18:8
```

   - 对一个常量进行赋值了
3. 什么时候用常量呢？
   - 如果有个值不会再发生改变的时候，我们可以使用常量存储

目录

Contents

04

数据类型

- 数据类型☆

4. 数据类型

为什么要对数据分类型？

生活中会把物品进行归类，不同的物品不能混淆在一起。

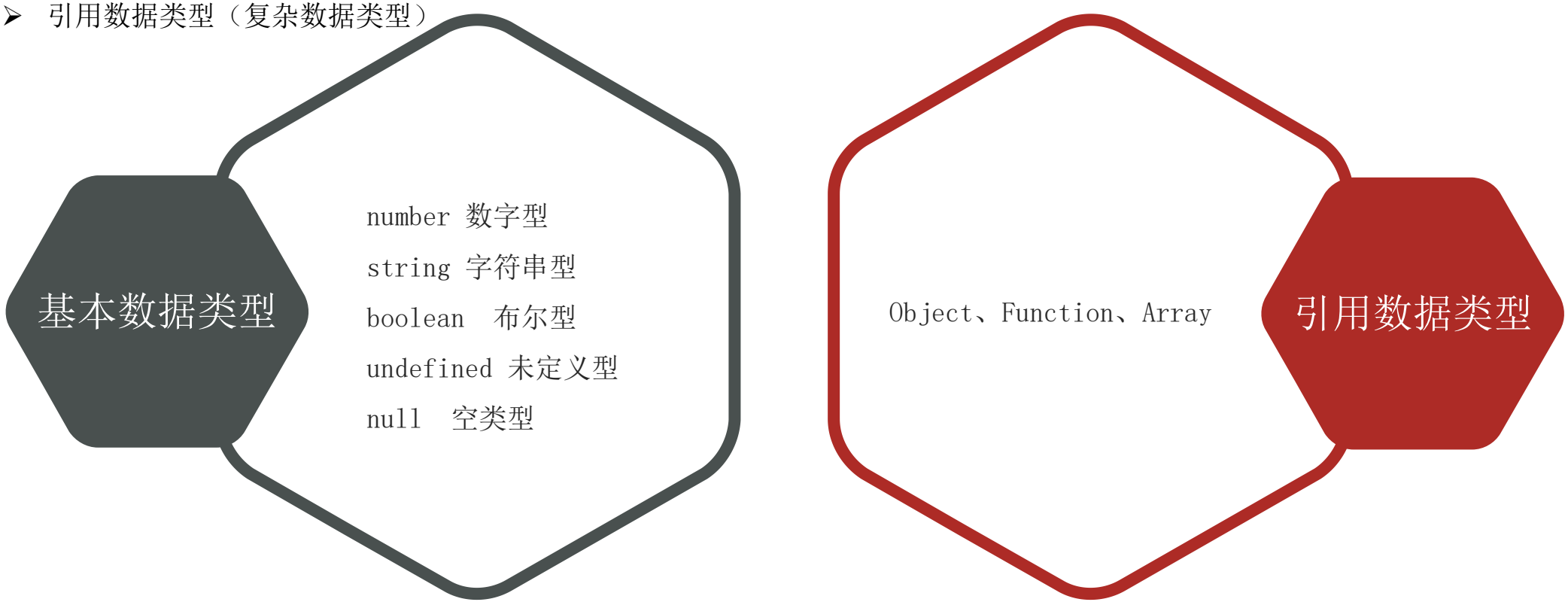
计算机程序可以处理大量的数据，方便程序员的使用数据。

比如：

4. 数据类型

JS 数据类型整体分为两大类：

- 基本数据类型（简单数据类型）
- 引用数据类型（复杂数据类型）



4. 基本数据类型 - 数字类型（Number）

数字：可以用于运算操作，比如对数字的加减乘除等操作

JavaScript 中的整数、小数等 统一称为 数字类型

```
let age = 18    // 整数
let price = 88.99 // 小数
```

通过 typeof 关键字检测数据类型

4. 数据类型

JS 数据类型整体分为两大类：

- 基本数据类型
- 引用数据类型

4. 基本数据类型 - 字符串类型（string）

字符串： 被引号包裹的一段文字信息

JS中的字符串：

通过单引号（ ‘’ ）、双引号（ “” ）或反引号（ `` ）包裹的数据都属于字符串

单引号和双引号没有本质上的区别，推荐使用单引号。

```
let uname = '小明'   // 单引号包含的数据是字符串
let sex = "男"       //  双引号包含的数据是字符串
let address = `黑马程序员`    // 反引号包含的数据是字符串
let tel = '13681221426'      // 虽然看上去是数字但是其实是字符串
```

注意事项：

1. 单引号/双引号可以互相嵌套，但是不以自己嵌套自己（口诀：外双内单，或者外单内双）
2. 一定注意变量名不要加引号，否则认为是字符串

总结

1. JS数据类型可以分为哪两大类？

- 基本数据类型（简单数据类型）
- 引用数据类型（复杂数据类型）

2. 数字型包含什么？数字类型的主要作用是啥？

- 整数和小数
- 用于计算的

3.什么是字符串数据类型？可以使用哪三种引号包裹？

- 用引号包裹的一段文字信息
- 单引号、双引号、反引号

4.可以通过哪个关键词检测数据类型呢 ？

- typeof

4. 基本数据类型 - 字符串类型（string）

反引号什么时候用呢?

是为了解决字符串拼接的问题，使用反引号（模板字符串）会非常方便

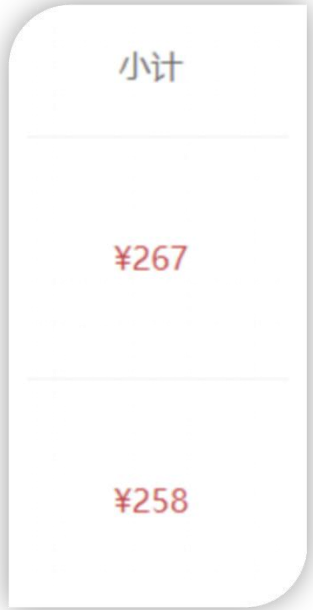
4. 基本数据类型- 字符串类型（string）

字符串拼接：

使用场景： + 可以实现字符串的拼接， 最常见是字符串拼接变量

口诀：数字相加，字符相连

```
// 1. 字符串拼接
let age = 19
console.log('pink老师今年' + age)
```



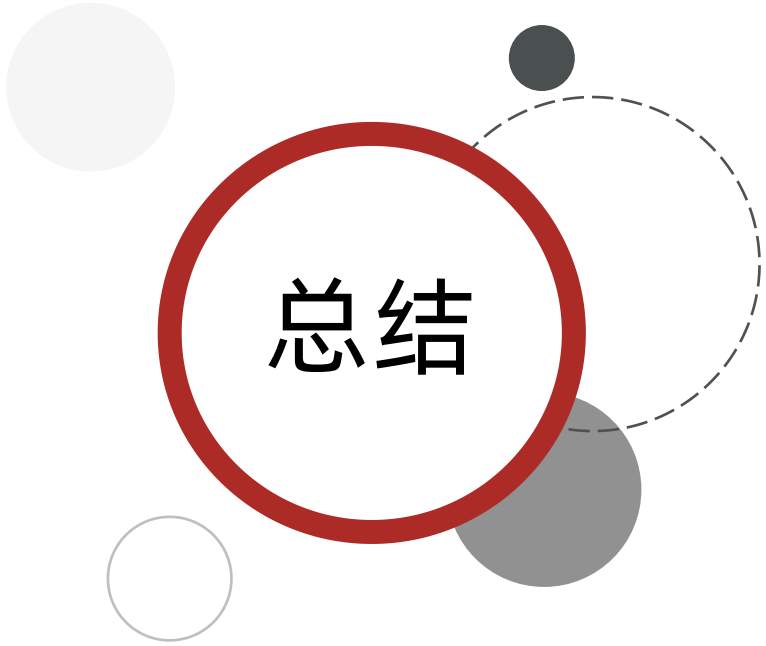
麻烦~

模板字符串（重点）

- 使用场景
  - 拼接字符串和变量
- 语法
  - ``（反引号）
  - 内容拼接变量时，用 ${ } 包住变量（口诀：反引中间变量套，直接$大括号）

```
console.log(`pink老师今年${age}岁`)
```

- 注意：
  - 反引号中间的字符串可以换行的



1. 用 + 字符串拼接比较麻烦，可以使用什么来解决这个问题？
   - 模板字符串， 可以让我们拼接字符串更简便
2. 模板字符串使用注意事项：
   - 用什么符号包含数据？
     - ✓ 反引号 ``
   - 用什么来使用变量？
     - ✓ ${变量名}
   - 口诀：反引中间变量套，直接$大括号
   - 模板字符串里面的字符可以换行的

```
document.write(`大家好，我叫${name}，今年${age}岁`)
```

案例

页面输出姓名和年龄案例

需求：页面弹出对话框，输入名字和年龄，页面显示： 大家好，我叫xxx，今年xx岁了

4. 数据类型

JS 数据类型整体分为两大类：

- 基本数据类型
- 引用数据类型

4. 基本数据类型- 布尔类型（boolean）

布尔型:用于判断真假的数据类型，通常用来判断条件是否成立

它有两个固定的值 true 和 false

- 表示肯定的数据用 true（真），表示否定的数据用 false（假）。

```
// JavaScript 好玩不?
let isCool = true
console.log(isCool)
```

4. 基本数据类型- 未定义类型（undefined）

未定义是比较特殊的类型，只有一个值 undefined。

什么情况出现未定义类型？

只声明变量，不赋值的情况下，变量的默认值为 undefined，一般很少【直接】为某个变量赋值为 undefined。

```
let age    // 声明变量但是未赋值
document.write(age)  // 输出 undefined
```

工作中的使用场景：

我们开发中经常声明一个变量，等待传送过来的数据。

如果我们不知道这个数据是否传递过来，此时我们可以通过检测这个变量是不是undefined，就判断用户是否有数据传递过来。

4. 基本数据类型- 空类型（null）

JavaScript 中的 null 仅仅是一个代表“无”、“空”或“值未知”的特殊值

```
let obj = null
console.log(obj)  // null
```

null 和 undefined 区别：

- undefined 表示没有赋值，不存在（ 期房）
- null 表示赋值了，但是内容为空 （毛坯房）

注意：

typeof null 返回的是 ‘object’ 返回的是对象类型

但这只是JavaScript 存在的一个悠久 Bug，不代表null就是引用数据类型,null 属于基本数据类型

总结

1. 用于判断真假的是哪种数据类型？有几种值？
   - 布尔型
   - true 和 false
2. 什么时候出现未定义数据类型？开发场景是？
   - 定义变量未给值就是 undefined
   - 如果检测变量是undefined就说明没有值传递过来
3. null 是什么类型？
   - 空类型

目录

Contents

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

5.1 算术运算符

数字是用来计算的，比如：乘法 * 、除法 / 、加法 + 、减法 - 等等，所以经常和算术运算符一起。

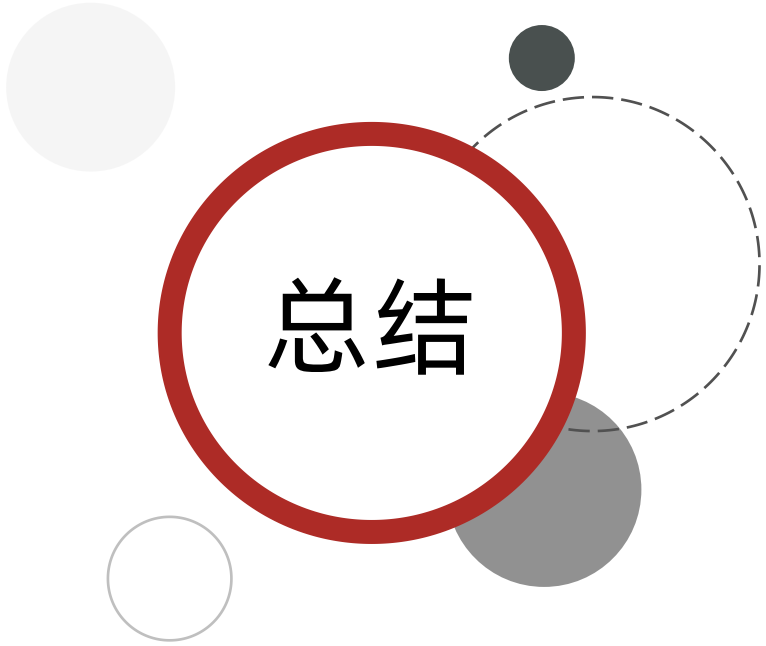
算术运算符：也叫数学运算符，主要包括加、减、乘、除、取余（求模）等

- +：求和
- -：求差
- *：求积
- /：求商
- %：取模（取余数）
  - 开发中经常作为某个数字是否被整除

注意：

在计算失败时，显示的结果是 NaN （not a number）

```
console.log('老师' - 2)  // NaN
```



1. 算术运算符有那几个常见的？
   - \+ - * / %
2. 取余运算符开发中的使用场景是？
   - % 来判断某个数字是否能被整除
3. 通常在计算失败时，显示的结果是？
   - NaN

案例

计算商品小计

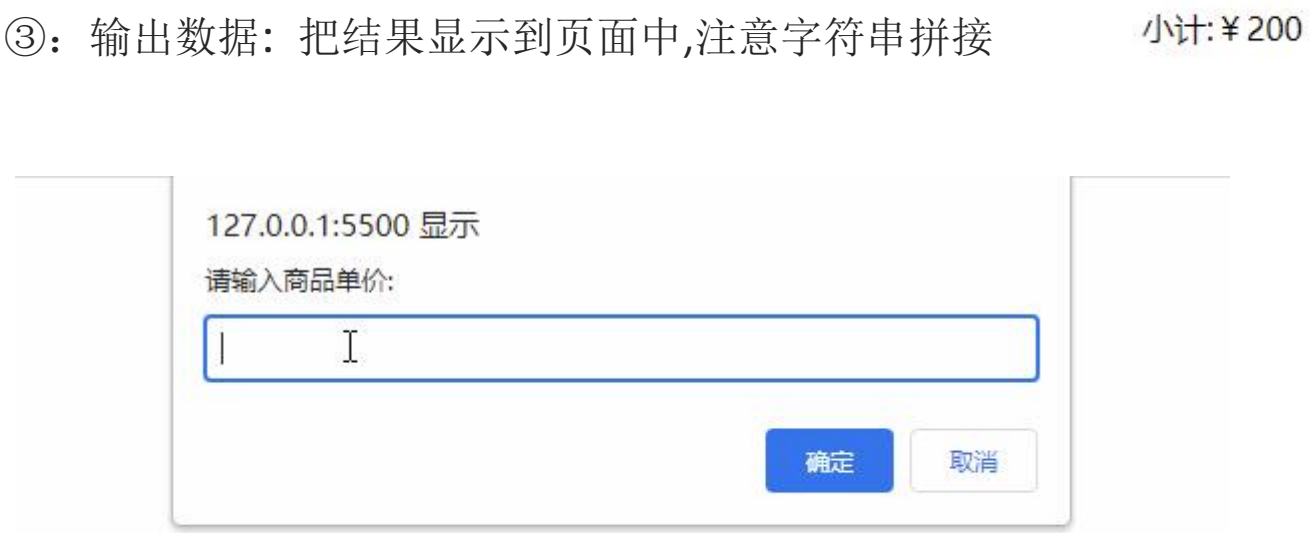
需求：用户输入商品价格和数量，可以计算出小计，显示到页面中

分析：

①：输入数据： 单价 price、数量 num

②：处理数据： 小计 = 单价 * 数量

③：输出数据：把结果显示到页面中,注意字符串拼接



05

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

5.2 赋值运算符

- 赋值运算符：对变量进行赋值的运算符
  - 已经学过的赋值运算符： =　　将等号右边的值赋予给左边，要求左边必须是一个容器

```
let num = 1
```

  - 其他赋值运算符：
    - +=
    - -=
    - *=
    - /=
    - %=
- 使用这些运算符可以在对变量赋值时进行快速操作，从而可以简化代码

5.2 赋值运算符

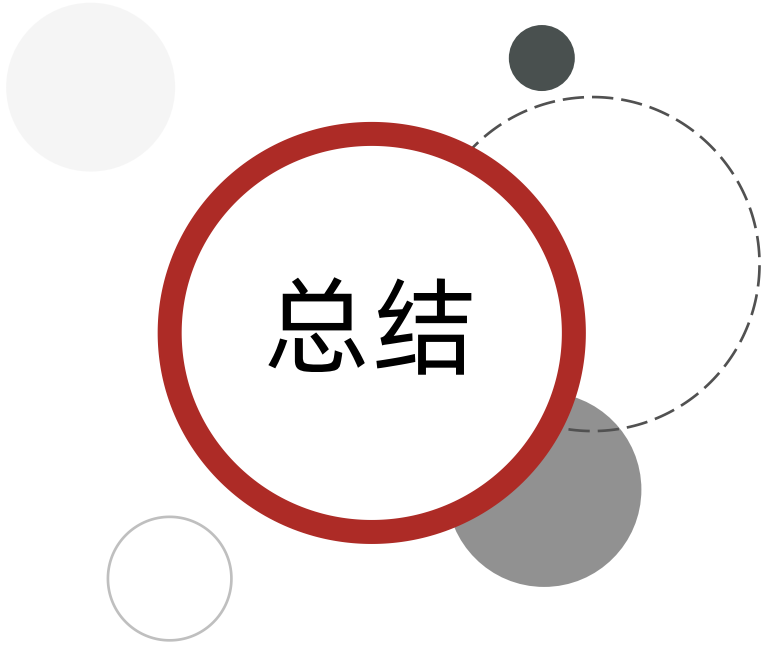
1. **以前**我们让一个变量加 1 如何做的？

```
<script>
    let num = 1
    num = num + 1
    console.log(num)  // 结果是 2
</script>
```

提问：想变量加3怎么写？

2. **现在**我们有一个简单的写法啦~~~

```
<script>
    let num = 1
    num += 1
    console.log(num)  // 结果是 2
</script>
```



1. = 赋值运算符执行过程？
   - 将等号右边的值赋予给左边，要求左边必须是一个容器
2. += 出现是为了简化代码，比如 let num = 10 ， 让 num 加5 怎么写呢？
   - num += 5

05

运算符

- 算术运算符
- 赋值运算符
- 自增/自减运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

5.3 自增/自减运算符

符号	作用	说明
++	自增	变量自身的值加1，例如：x++
--	自减	变量自身的值减1，例如：x--

注意事项

1. 只有变量能够使用自增和自减运算符
2. ++、-- 可以在变量前面也可以在变量后面，比如：x++ 或者 ++x

5.3 自增/自减运算符

```
let x = 3
++x
console.log(x)  // 4
```

单独使用的时候，++在前和在后没有区别，但是如果要参与运算就有区别了。

++、--放在变量前后的区别：（了解）

- ++放在变量前面，先对变量值进行+1，再拿变量的值进行运算
- 前缀式

```
// 前缀式
let x = 3
let y = ++x
```

- ++放在变量后面，先拿变量值进行运算，再对变量的值进行+1
- 后缀式

```
// 后缀式
let x = 3
let y = x++
```

5.3 自增/自减运算符

```
let x = 3
x++    // 当前类似于 x = x + 1  或者  x += 1
```

经验：

1. ++在前和++在后在单独使用时二者并没有差别，而且一般开发中我们都是独立使用
2. ++在后（后缀式） 我们会使用更多

总结

1.自增、自减运算符是什么？作用是什么？

- ++ --
- 可以让变量自身的值加1、减1

2. 实际开发中，我们一般都是单独使用的，++在后（后缀式）更多

05

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

5.4 比较运算符（关系运算符）

- 比较运算符
  - 使用场景：比较两个数据大小、是否相等，根据比较结果返回一个布尔值（true / false）

5.4 比较运算符

- 比较运算符：
  - > 左边是否大于右边
  - < 左边是否小于右边
  - >= 左边是否大于或等于右边
  - <= 左边是否小于或等于右边
  - === 左右两边是否**类型**和**值**都相等
  - == 左右两边**值**是否相等
  - != 左右值不相等
  - !== 左右两边是否不全等
- 结果：
  - 比较结果为布尔类型，即只会得到 true 或 false
- 对比：
  - = 是赋值
  - === 是全等
  - == 是判断

05

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

5.5 逻辑运算符

- **使用场景：**可以把多个布尔值放到一起运算，最终返回一个布尔值

5.5 逻辑运算符

- 逻辑运算符

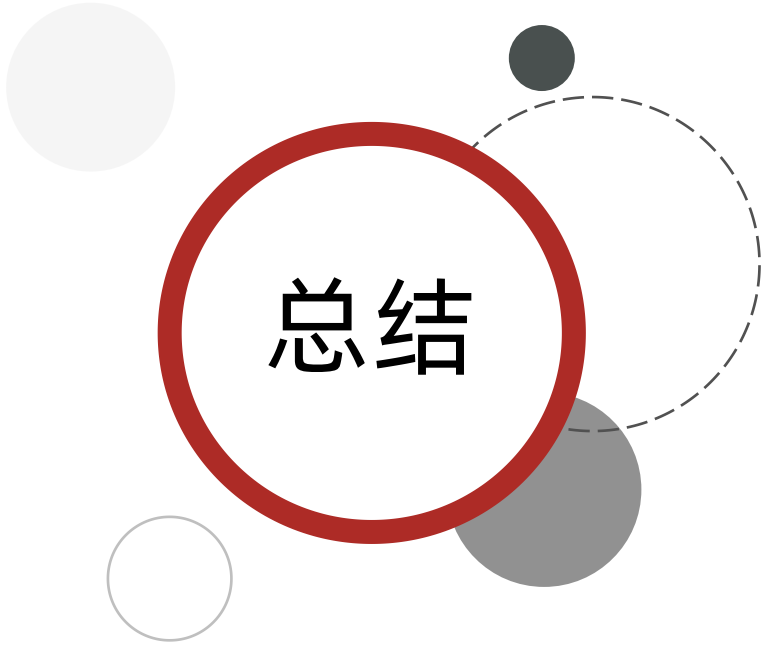
符号	名称	日常读法	特点	口诀
&&	逻辑与	并且	符号两边有一个假的结果为假	一假则假
||	逻辑或	或者	符号两边有一个真的结果为真	一真则真
!	逻辑非	取反	true变false false变true	真变假，假变真

5.5 逻辑运算符

- 逻辑运算符

A	B	A && B	A || B	!A
false	false	false	false	true
false	true	false	true	true
true	false	false	true	false
true	true	true	true	false

一假则假　　一真则真



1. 逻辑运算符有哪三个？
   - 与(&&)　或(||)　非(!)
2. 逻辑与怎么记忆呢？逻辑或怎么记忆呢？
   - 逻辑与：一假则假
   - 逻辑或：一真则真
3. 判断一个变量 num 是否大于5且小于10怎么写？

   num > 5 && num < 10

案例

判断一个数是4的倍数，且不是100的倍数

需求：用户输入一个，判断这个数是4的倍数，并且不是100的倍数，如果是则页面弹出true，否则弹出 false

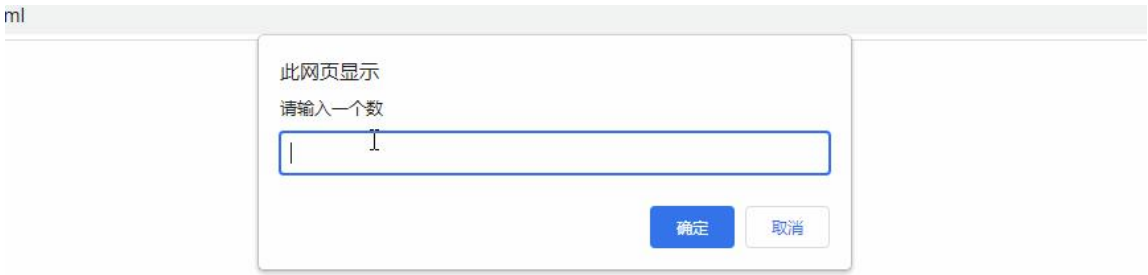
分析：

①：输入数据

②：处理数据

（1）判断条件： 倍数怎么判断 能被整除 → 余数是 0

③：输出结果



05

运算符

- 算术运算符
- 赋值运算符
- 一元运算符
- 比较运算符
- 逻辑运算符
- 运算符优先级

5.6 运算符优先级

优先级	顺序
1	()
2	++ -- !
3	先 * / % 后 + -
4	> >= < <=
5	== != === !==
6	先 && 后 ||
7	=

- 逻辑运算符优先级：！ > && > ||

5.6 运算符优先级

目标：掌握运算符优先级，能判断运算符执行的顺序

```
let n = 2000
console.log((n % 4 === 0) && (n % 100 !== 0) || (n % 400 === 0))
```

```
// 输出结果是什么?
let n = 2000
console.log(n % 4 === 0 && n % 100 !== 0 || n % 400 === 0) // ?
```

描述	单词	效果
输入输出语句	alert('页面弹出警示框')	输出语句，页面弹出警示框
	document.write('页面打印输出')	输出语句，页面文档打印输出
	console.log('控制台打印输出')	输出语句，控制台输出
	prompt('请输入姓名:')	输入语句，输入对话框
声明变量	let num = 10	声明变量
声明常量	const pi = 3.14	声明常量
基本数据类型	number	数字类型
	string	字符串类型
	boolean	布尔类型
	undefined	未定义类型
	null	空类型
运算符	算术运算符、比较运算符、逻辑运算符、一元运算符、赋值运算符	

目录

Contents

案例

综合案例-用户订单详情案例

需求：用户输入商品价格和商品数量，以及收货地址，页面显示商品订单信息表格

订单详情页面

商品名称	商品价格	商品数量	总价	收货地址
小米青春版PLUS	1999元	2	3998元	黑马程序员

案例

用户订单信息案例

需求：用户输入商品价格和商品数量，以及收货地址，可以自动打印订单信息

分析：

①：输入数据

需要输入3个数据，所以需要3个变量来存储　价格 price、数量 num、地址 address

②：处理数据

需要计算总的价格保存变量：总计 total = price * num

③：输出数据

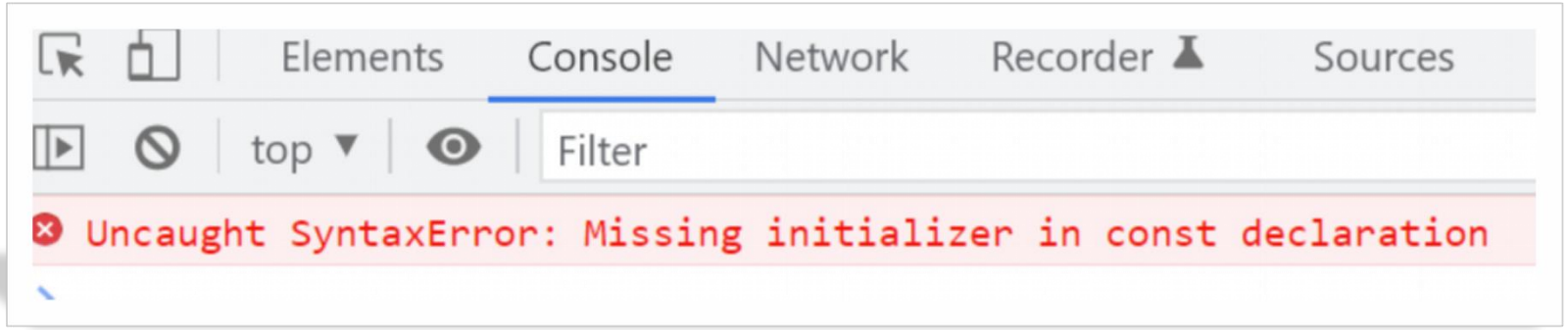
页面打印生成表格，里面填充数据，使用模板字符串

订单详情页面

商品名称	商品价格	商品数量	总价	收货地址
小米青春版PLUS	1999元	2	3998元	黑马程序员

六. 常见错误

- 下面可能出现的原因是什么？



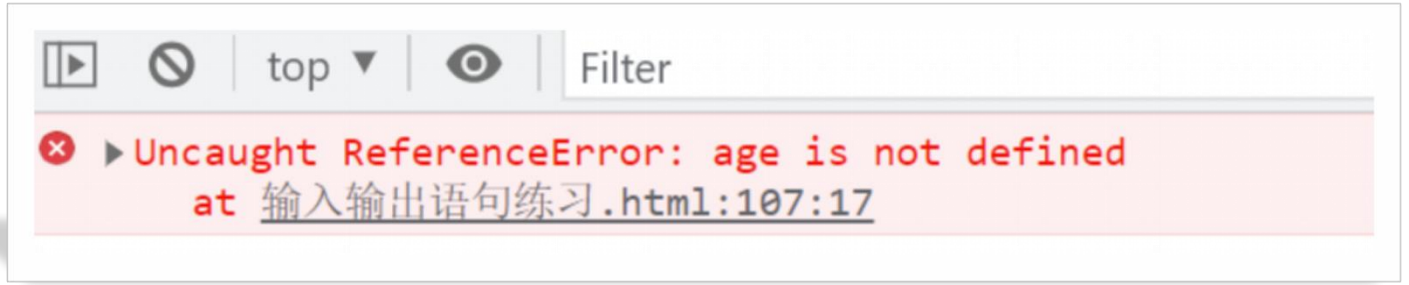

```
const age
console.log(age)  // 错误 const 必须有赋值操作
```

分析：

- const 声明的时候必须要赋值，进行初始化

六. 常见错误

- 下面可能出现的原因是什么？



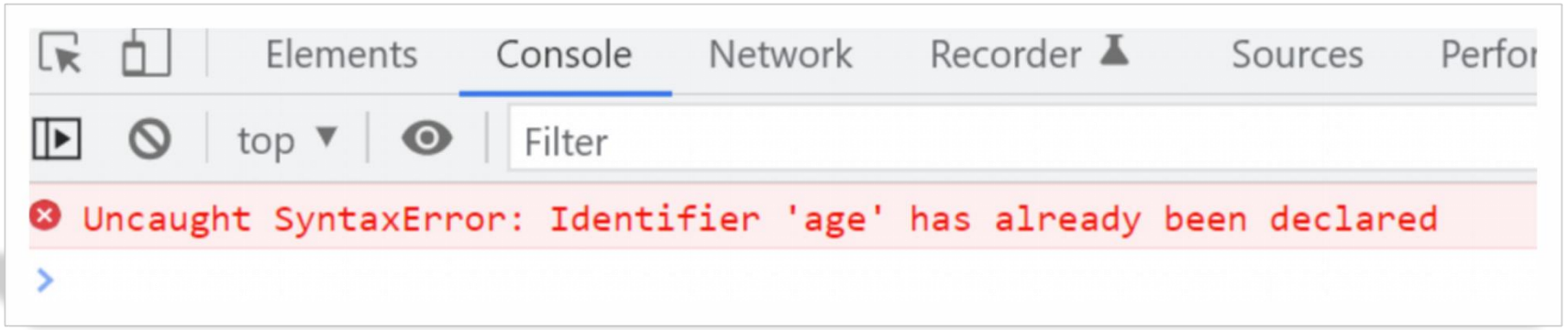

```
console.log(age)  // 么有age变量
```

分析：

- 提示 age变量没有定义过
- 很可能 age 变量没有声明和赋值
- 或者我们输出变量名和声明的变量不一致引起的（简单说写错变量名了）

六. 常见错误

- 下面可能出现的原因是什么？



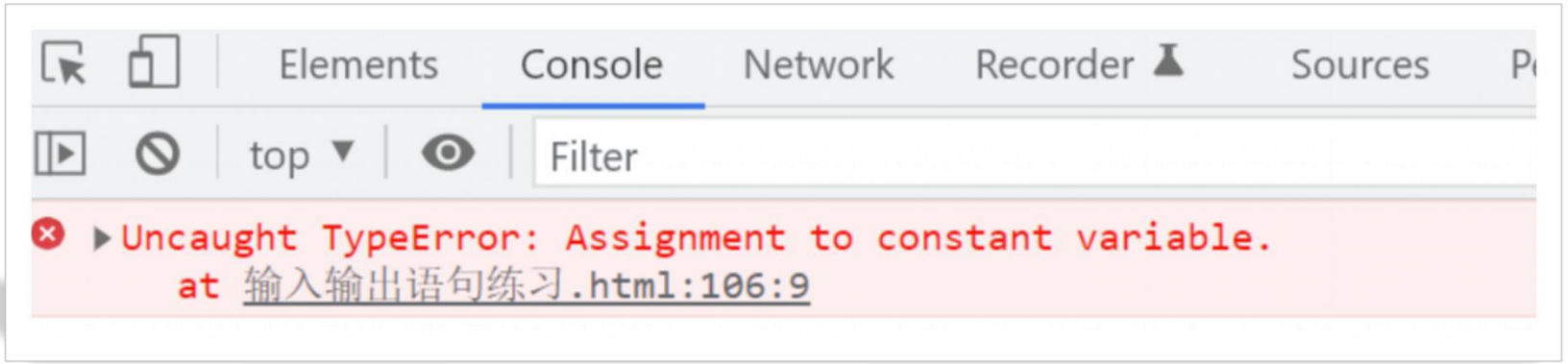
分析：

- 提示 “age”已经声明
- 很大概率是因为重复声明了一个变量。
- 注意let 或者const 不允许多次声明同一个变量

```
let age = 18
let age = 21
console.log(age)  //  错误 let不允许重复声明
```

六. 常见错误

- 下面可能出现的原因是什么？



分析：

- 常量被重新赋值了
- 常量不能被重新赋值

```
const age = 18
age = 21
console.log(age)  //  错误 常量不能被重新赋值
```

传智教育旗下高端IT教育品牌