# JavaScript 基础第一天

基础语法

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

1. 理解变量是存储数据的“容器”
2. 理解什么是数据并知道数据的分类
3. 知道 JavaScript 数据类型转换的特征

# 目录

Contents


# JavaScript 介绍

- JavaScript 是什么
- JavaScript 书写位置
- JavaScript 的注释
- JavaScript的结束符
- 输入和输出语法
- 字面量

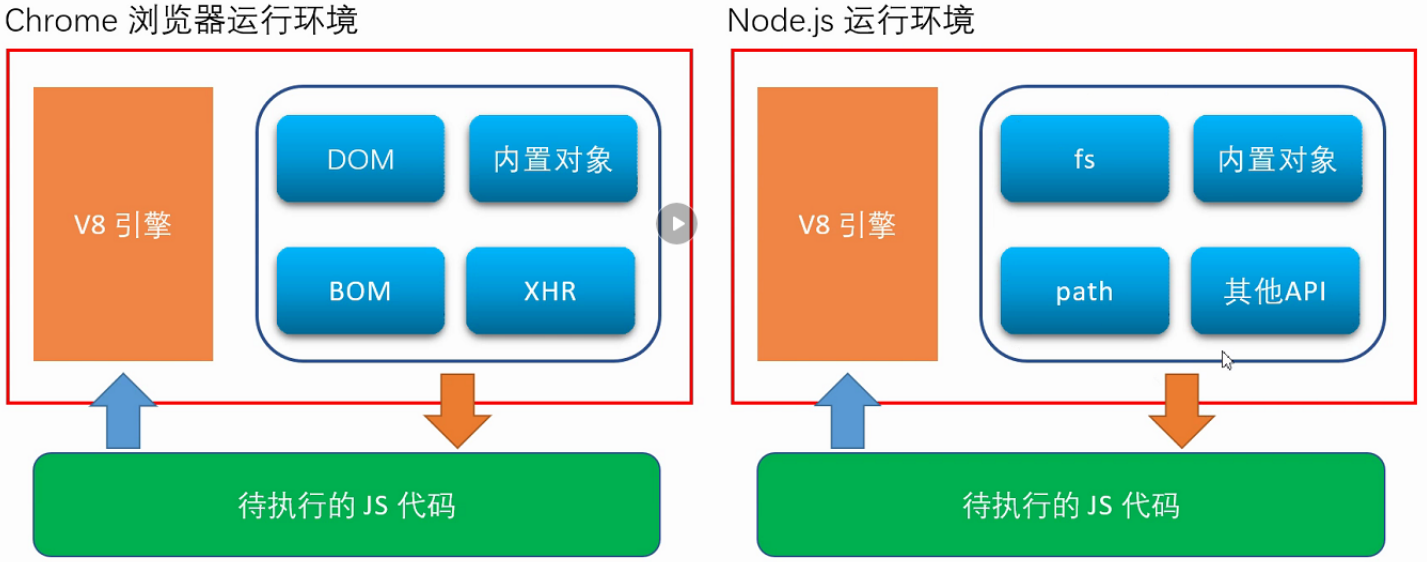
# 1.1 JavaScript 是什么

1. JavaScript （是什么？）

是一种运行在客户端（浏览器）的编程语言，实现人机交互效果。

2. 作用（做什么？）

- 网页特效 (监听用户的一些行为让网页作出对应的反馈)
- 表单验证 (针对表单数据的合法性进行判断)
- 数据交互 (获取后台的数据, 渲染到前端
- 服务端编程 (node.js)



# 1.1 JavaScript 是什么

## 3. JavaScript的组成（有什么？）

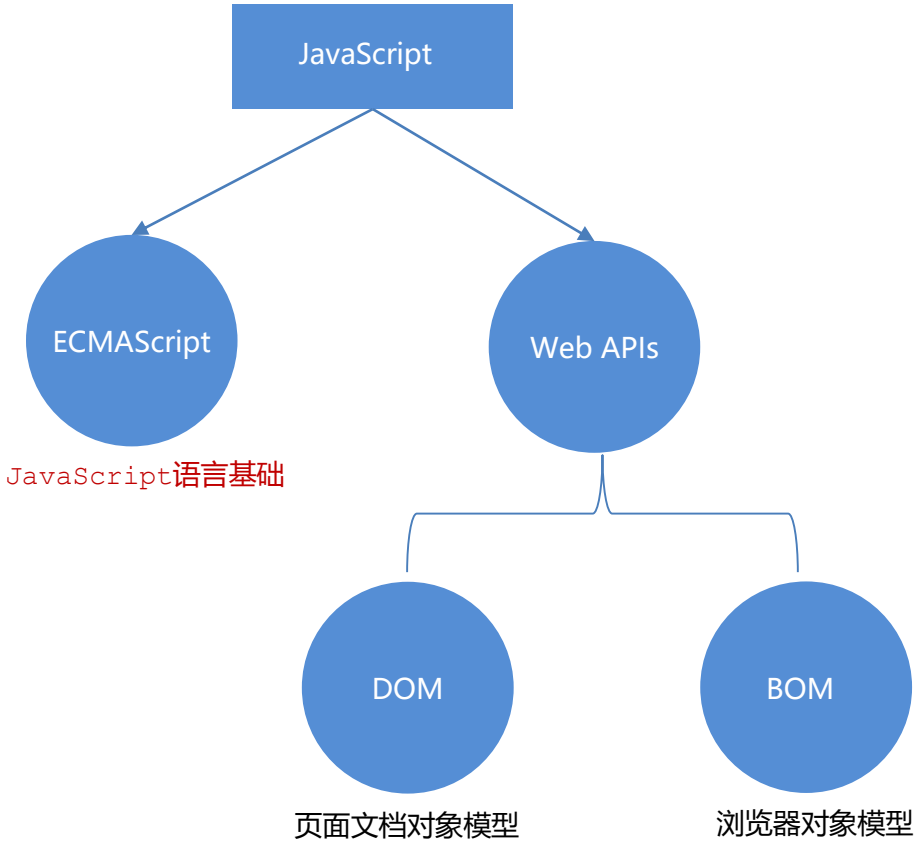
- ECMAScript:

  规定了js基础语法核心知识。

  - 比如：变量、分支语句、循环语句、对象等等

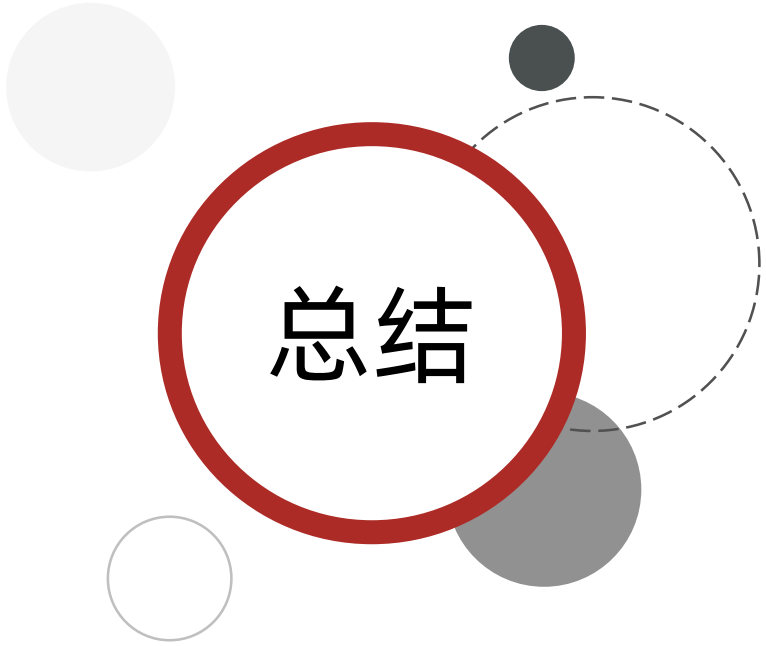
- Web APIs :

  - DOM 操作文档，比如对页面元素进行移动、大小、添加删除等操作
  - BOM 操作浏览器，比如页面弹窗，检测窗口宽度、存储数据到浏览器等等




**权威网站： MDN**

**JavaScript权威网站：** https://developer.mozilla.org/zh-CN/docs/Web/JavaScript

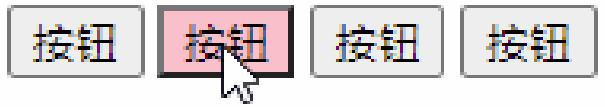



1. JavaScript是什么？
   - JavaScript 是一种运行在客户端（浏览器）的编程语言
2. JavaScript组成是什么？
   - ECMAScript( 基础语法 )、web APIs （DOM、BOM）

# 思路

## 体验-JavaScript

点击切换按钮的案例-体验HTML+CSS+JS 实现交互效果

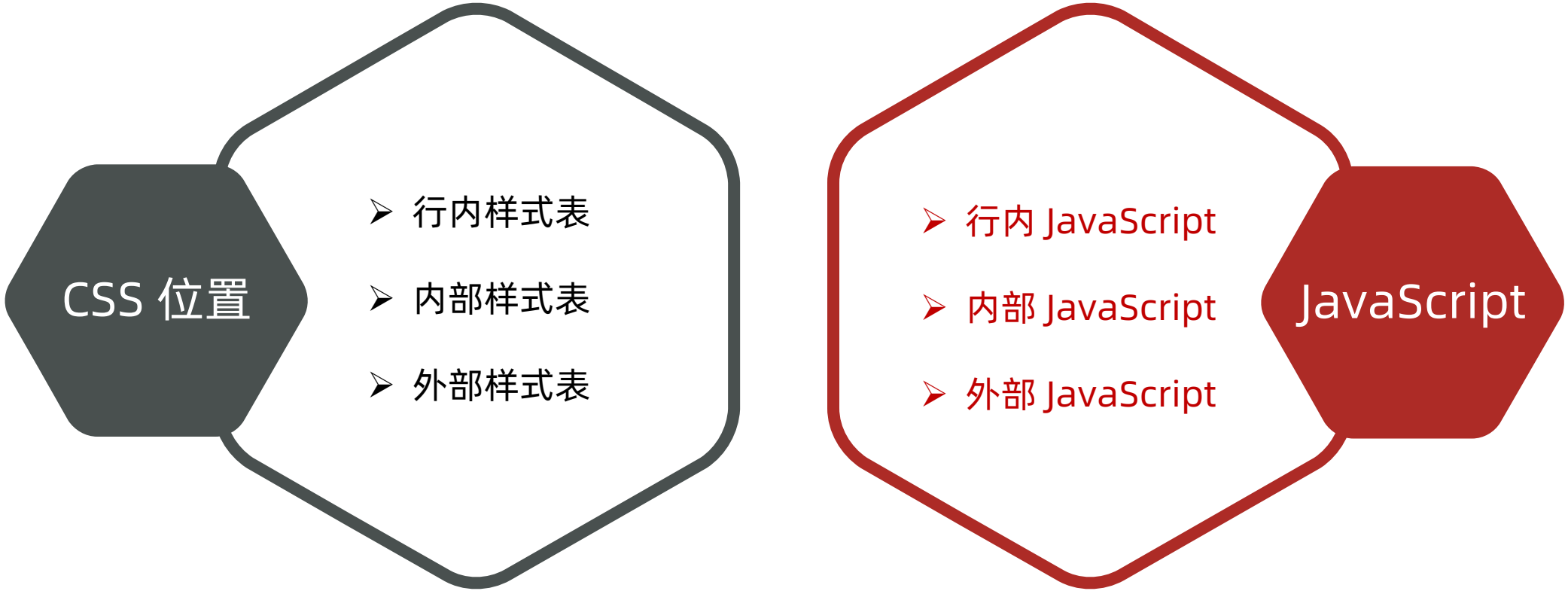



# JavaScript 介绍

- JavaScript 是什么
- JavaScript 书写位置
- JavaScript 的注释
- JavaScript的结束符
- 输入和输出语法
- 字面量

## 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，js') 页面弹出警告对话框

```
<script>
    alert('嗨，欢迎来传智播学习前端技术！')
</script>
</body>
```

**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

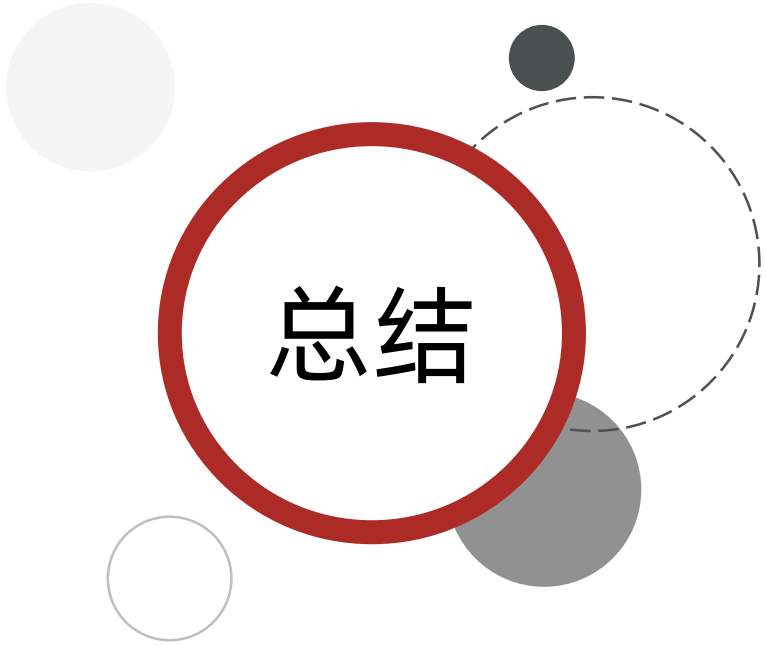
### 3. 内联 JavaScript

代码写在标签内部

**语法：**

注意： 此处作为了解即可，但是后面vue框架会用这种模式

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




1. JavaScript三种书写位置？
   - 内部
   - 外部
   - 行内
2. 注意事项：
   - 书写的位置尽量写到文档末尾 </body> 前面
   - 外部 js 标签中间不要写代码，否则会被忽略

练习

## 页面弹框课堂练习

需求：请用外部和内部两种 JavaScript 书写方式，页面弹出： 努力，奋斗

时间：5分钟

举例说明

页面弹窗： alert(‘xxxx’)


# JavaScript 介绍

- JavaScript 是什么
- JavaScript 书写位置
- JavaScript 的注释
- JavaScript的结束符
- 输入和输出语法
- 字面量

# 1.3 JavaScript 注释

目标：会使用两种JavaScript注释方法

- 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>
```


# JavaScript 介绍

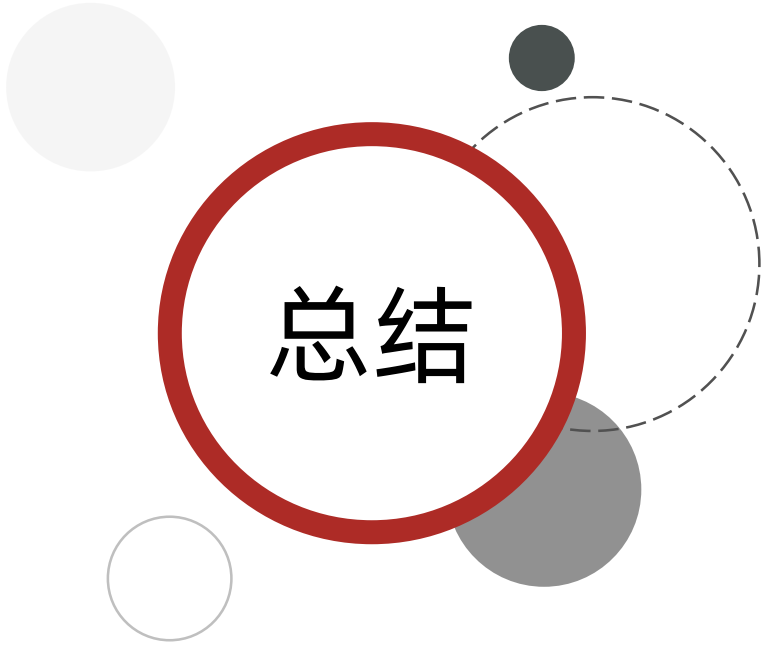
- JavaScript 是什么
- JavaScript 书写位置
- JavaScript 的注释
- JavaScript的结束符
- 输入和输出语法
- 字面量

# 1.4 JavaScript 结束符

- 目标：了解JavaScript结束符
- 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. JavaScript 注释有那两种方式？
   - 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


# JavaScript 介绍

- JavaScript 是什么
- JavaScript 书写位置
- JavaScript 的注释
- JavaScript的结束符
- 输入和输出语法
- 字面量

# 1.5 JavaScript 输入输出语法

目标：能写出常见 JavaScript 输入输出语法

**什么是语法：**

- 人和计算机打交道的规则约定
- 我们要按照这个规则去写
- 比如： 你吃了吗？
- 我们程序员需要操控计算机，需要计算机能看懂

# 1.5 JavaScript 输入输出语法

目标：能写出常见 JavaScript 输入输出语法

输出和输入也可理解为人和计算机的交互，用户通过键盘、鼠标等向计算机输入信息，计算机处理后再展示结果给用户，这便是一次输入和输出的过程。

- 输出语法：

**语法1：**

```
document.write('要出的内容')
```

**作用：** 向body内输出内容

**注意：** 如果输出的内容写的是标签，也会被解析成网页元素

**语法2：**

```
alert('要出的内容')
```

**作用：** 页面弹出警告对话框

**语法3：**

```
console.log('控制台打印')
```

**作用：** 控制台输出语法，程序员调试使用

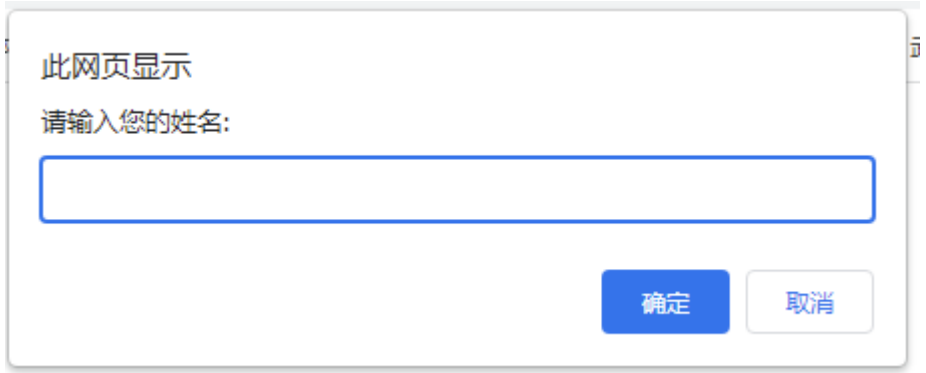
## 1.5 JavaScript 输入输出语法

2. 输入语法：

- **语法：**

```
prompt('请输入您的姓名:')
```

- **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：**

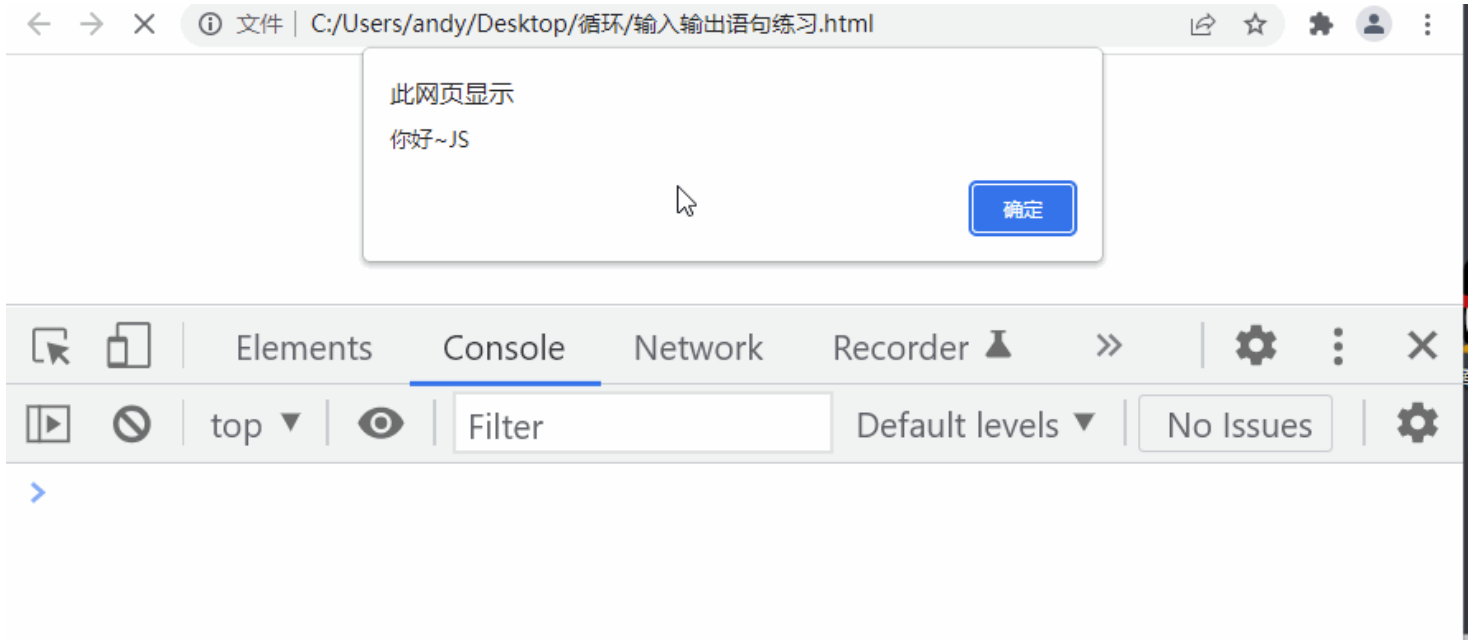


练习

## 输入和输入练习

需求：时间：5分钟

- 浏览器中弹出对话框： 你好 JS~
- 页面中打印输出： JavaScript 我来了!
- 页面控制台输出： 它~会魔法吧~



# 1.5 JavaScript 输入输出语法

**JavaScript 代码执行顺序：**

- 按HTML文档流顺序执行JavaScript代码
- alert() 和 prompt() 它们会跳过页面渲染先被执行（目前作为了解，后期讲解详细执行过程）


# JavaScript 介绍

- JavaScript 是什么
- JavaScript 书写位置
- JavaScript 的注释
- JavaScript的结束符
- 输入和输出语法
- 字面量

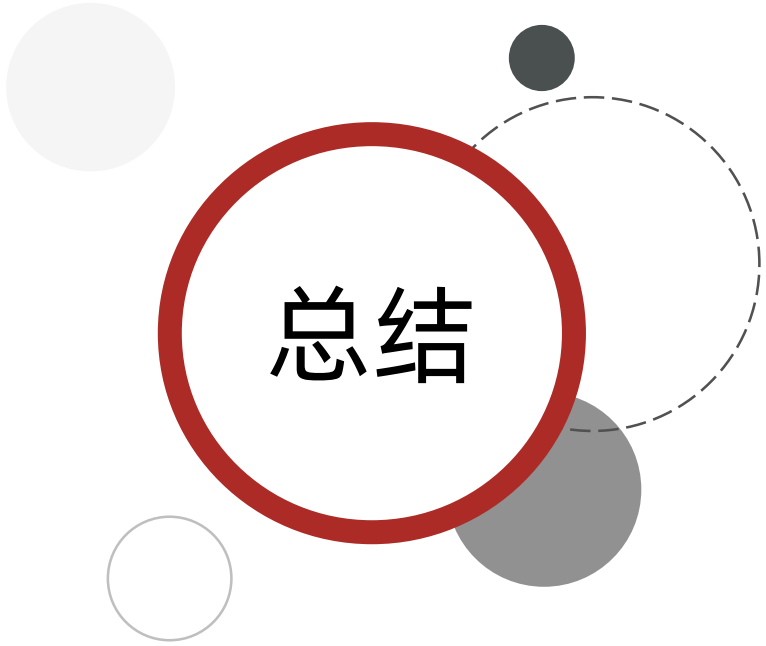
## 1.6 字面量

目标：能说出什么是字面量

在计算机科学中，字面量（literal）是在计算机中描述 事/物

比如：

- 我们工资是： 1000 此时 1000 就是 数字字面量
- '黑马程序员' 字符串字面量
- 还有接下来我们学的 [] 数组字面量 {} 对象字面量 等等




1. JavaScript是什么？

JavaScript是一门编程语言，可以实现很多的网页交互效果。

2. JavaScript 书写位置？

- 内部 JavaScript
- 内部 JavaScript - 写到 </body> 标签上方
- 外部 JavaScript - 但是 <script> 标签不要写内容，否则会被忽略

3. JavaScript 的注释？

- 单行注释 //
- 多行注释 /* */

4. JavaScript 的结束符？

- 分号； 可以加也可以不加，可以按照团队约定

5. JavaScript 输入输出语句？

- 输入： prompt()
- 输出： alert() document.write() console.log()

# 目录

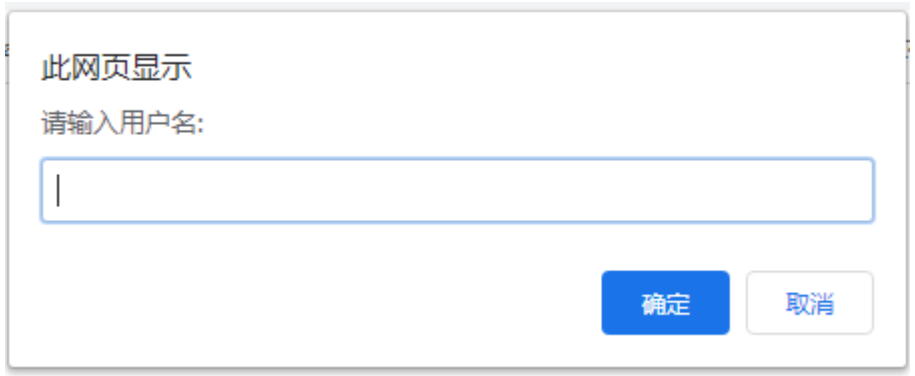
Contents

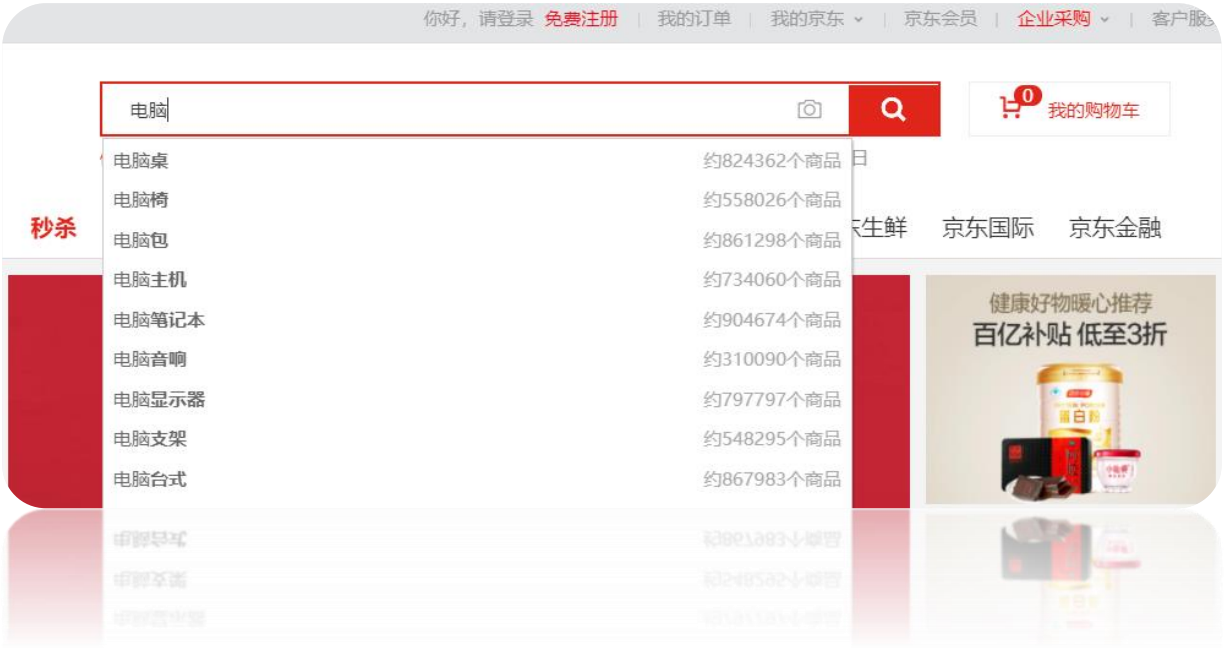

# 变量

- 变量是什么
- 变量基本使用☆
- 变量的本质
- 变量命名规则与规范

## 2.1 变量是什么？

问题1：用户输入的数据我们如何存储起来？



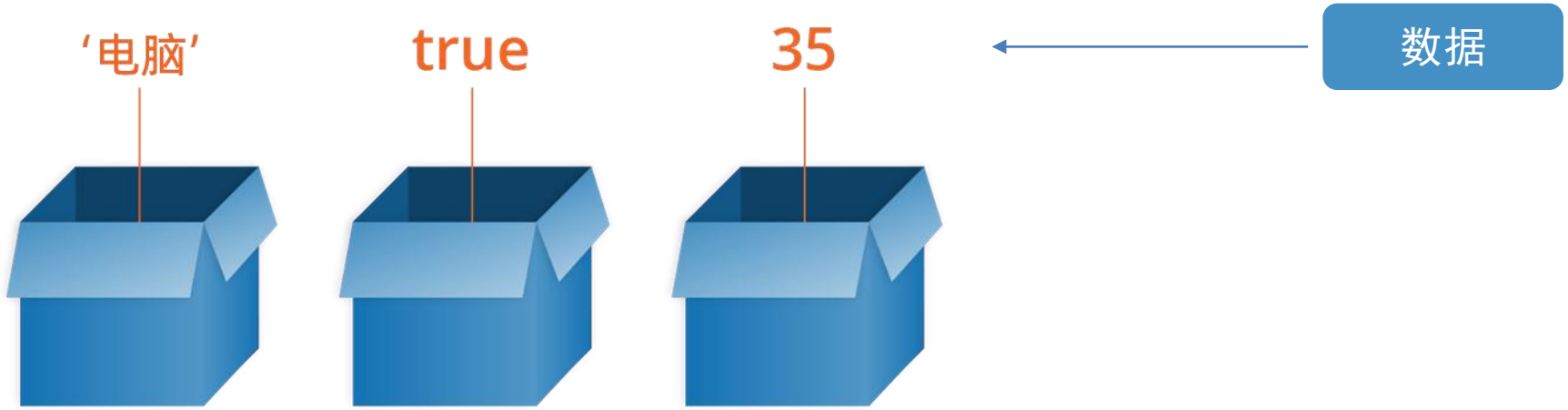

答案1：变量

## 2.1 变量是什么？

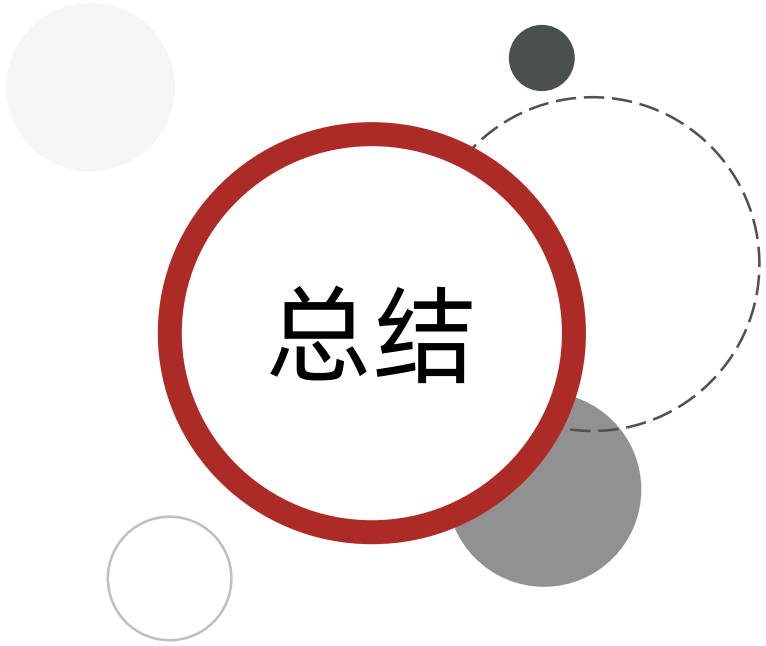
目标：理解变量是计算机存储数据的“容器”

1. **变量：**

- 白话：变量就是一个装东西的盒子。
- 通俗：变量是计算机中用来**存储数据**的“容器”，它可以让计算机变得有记忆。




- 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。




1. 变量是怎么理解？
   - 计算机中用来存储数据的“容器”，简单理解是一个个的盒子。
2. 变量有什么作用呢？
   - 用来存放数据的。注意变量指的是容器而不是数据。
3. 你还能想到那些生活中的变量？
   - HTML标签
   - 教室
   - 宿舍
   - ...

02

# 变量

- 变量是什么
- 变量基本使用☆
- 变量的本质
- 变量命名规则与规范

## 2.2 变量的基本使用

目标：能够声明一个变量并完成赋值操作

1. 变量的声明
2. 变量的赋值

## 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. 变量用什么关键字来声明？
   - let
2. 变量通过什么符号来赋值？
   - = 这个符号我们也称为 赋值运算符
3. 开发中我们经常声明的同时可以直接赋值？

```
<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

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

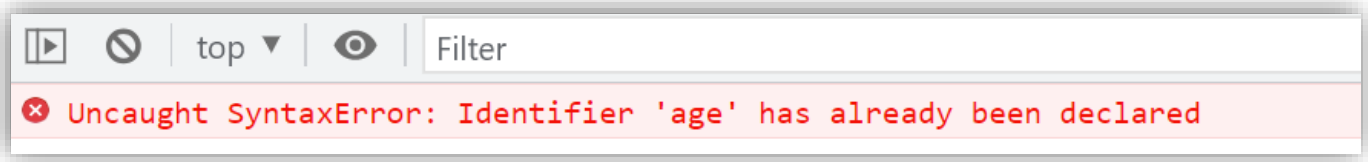
**3. 更新变量：**

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，同时里面存放了 18 这个数据
let age = 18
// 这里不允许多次声明变量
let age = 19
// 输出会报错
document.write(age)
```

注意： let 不允许多次声明一个变量。



## 2.2 变量的基本使用

**4. 声明多个变量：**

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。

**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，请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. 我们提倡同时声明多个不同的变量吗？
   - 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'
```

案例

## 1. 变量案例-弹出姓名

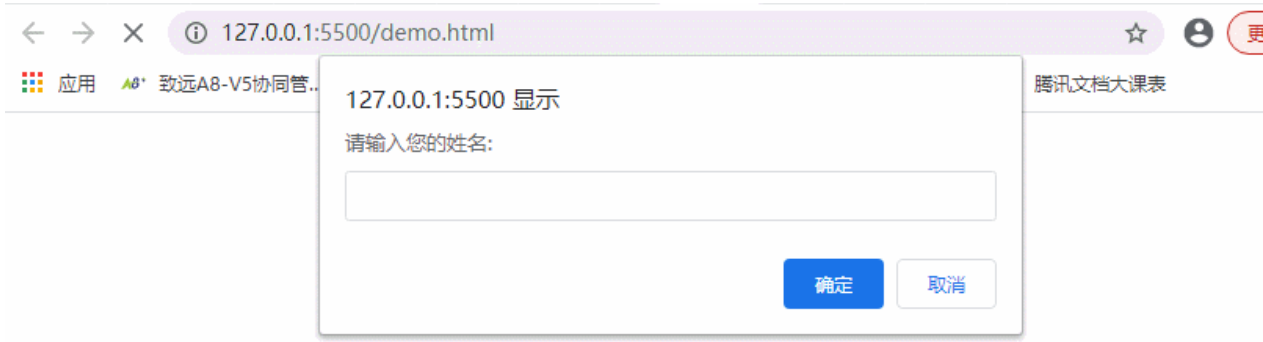
需求： 浏览器中弹出对话框： 请输入姓名，　　页面中输出：刚才输入的姓名

分析：

①：输入：用户输入框：prompt()

②：内部处理：保存数据

③：输出： 页面打印 document.write()



案例

## 2.变量案例- 交换变量的值

需求：

有2个变量： num1 里面放的是 10， num2 里面放的是20

最后变为 num1 里面放的是 20 ， num2 里面放的是 10

目的：

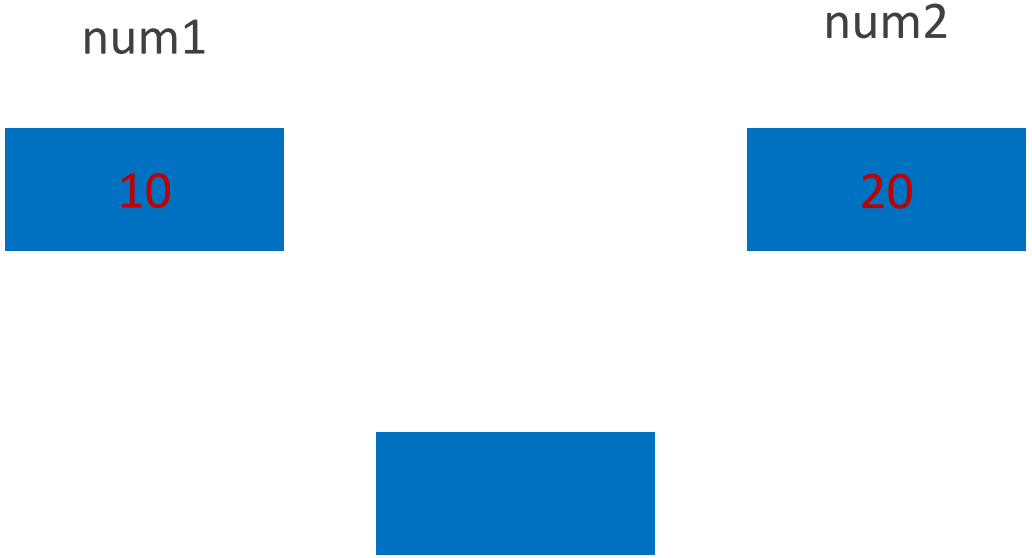
1. 练习变量的使用
2. 为了后面冒泡排序做准备

案例

## 2.变量案例- 交换变量的值

分析：

1.核心思路：使用一个 临时变量 用来做中间存储




步骤：

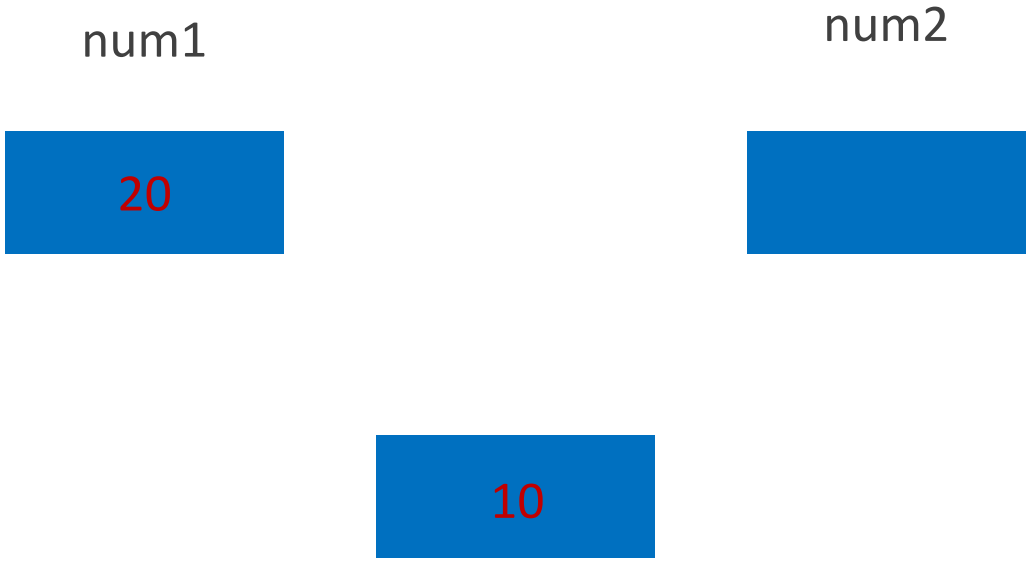
1. 声明一个临时变量 temp
2. 把num1的值赋值给 temp
3. 把num2的值赋值给num1

案例

## 2.变量案例- 交换变量的值

分析：

1.核心思路：使用一个 临时变量 用来做中间存储




步骤：

1. 声明一个临时变量 temp
2. 把num1的值赋值给 temp
3. 把num2的值赋值给num1
4. 把temp的值给num2

没了~~~~临时变量不用自动销毁

02

# 变量

- 变量是什么
- 变量基本使用☆
- 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


# 变量

- 变量是什么
- 变量基本使用☆
- 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：能写出符合规范的变量名

**规则：**必须遵守，不遵守报错（法律层面）

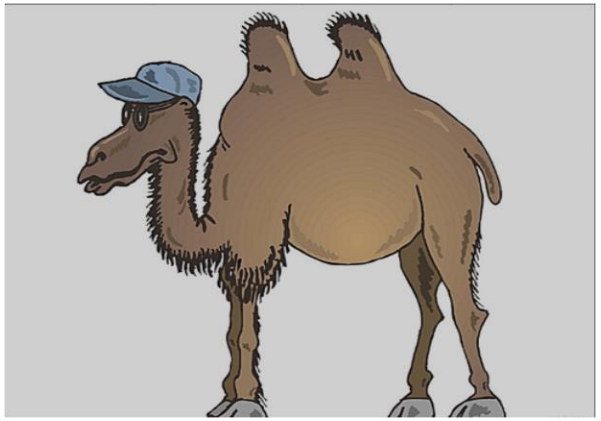
**规范：**建议，不遵守不会报错，但不符合业内通识 （道德层面）

**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



## 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 | | |
| _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

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(gender)： 用变量保存起来。

④：页面分别 输出 (document.write) 刚才的 3 个变量。

## 二. 变量拓展-let和var的区别

**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：**

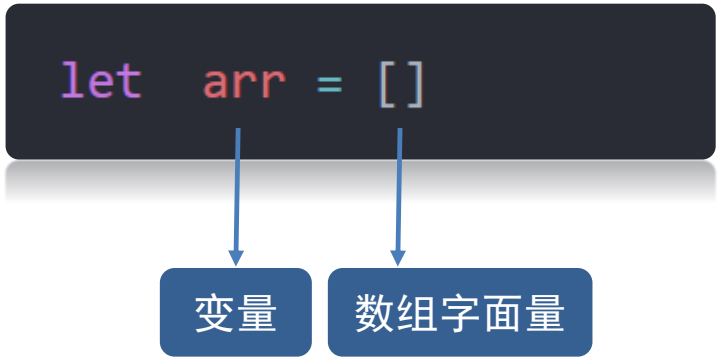
var 就是个bug，别迷恋它了，以后声明变量我们统一使用 let


1. 变量一次只能存几个值？

2. 如果我想保存一个班里5个人的姓名怎么办？

## 二. 变量拓展-数组

- 数组 (Array) —— 一种将 **一组数据存储在单个变量名下** 的优雅方式



## 1.1 数组的基本使用

目标：能够声明数组并且能够获取里面的数据

- **1. 声明语法**

```
let 数组名 = [数据1, 数据2, ..., 数据n]
```

- 例

```
let names = ['小明', '小刚', '小红', '小丽', '小米']
```

- 数组是按顺序保存，所以每个数据都有自己的编号
- 计算机中的编号从0开始，所以小明的编号为0，小刚编号为1，以此类推
- 在数组中，数据的编号也叫**索引或下标**
- 数组可以存储任意类型的数据

## 1.1 数组的基本使用

- **2. 取值语法**

```
数组名[下标]
```

- 例

```
let names = ['小明', '小刚', '小红', '小丽', '小米']
names[0]  // 小明
names[1]  // 小刚
```

- 通过下标取数据
- 取出来是什么类型的，就根据这种类型特点来访问

案例

## 数组取值案例

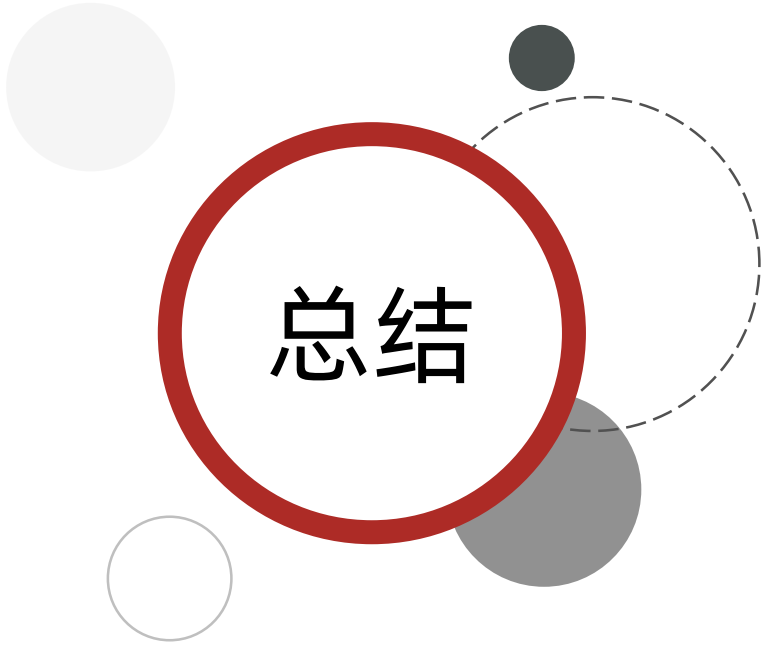
需求：定义一个数组，里面存放星期一、星期二..... 直到星期日（共7天），在控制台输出：星期日

# 1.1 数组的基本使用

**3. 一些术语：**

- **元素：**数组中保存的每个数据都叫数组元素
- **下标：**数组中数据的编号
- **长度：**数组中数据的个数，通过数组的length属性获得

```
let names = ['小明', '小刚', '小红', '小丽', '小米']
console.log(names[0])  // 小明
console.log(names[1]) // 小刚
console.log(names.length) //  5
```




1. 使用数组有什么好处？
   - 数组可以保存多个数据
2. 数组字面量用什么表示？
   - [] 中括号
3. 请说出下面数组中‘小米’的下标是多少？ 如何取得这个数据？
   - 下标是 4
   - 获取的写法是 names[4]

```
let names = ['小明', '小刚', '小红', '小丽', '小米']
```

# 目录

Contents


# 常量

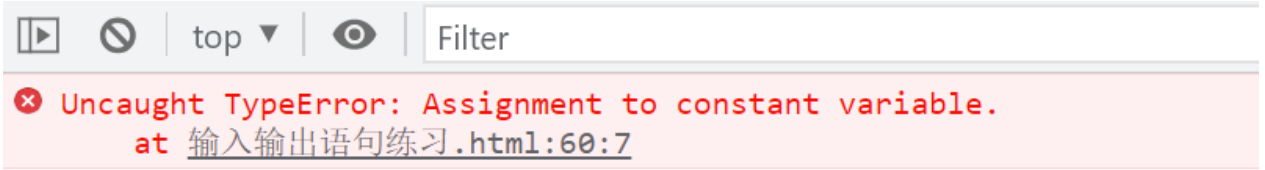
## 3. 常量的基本使用

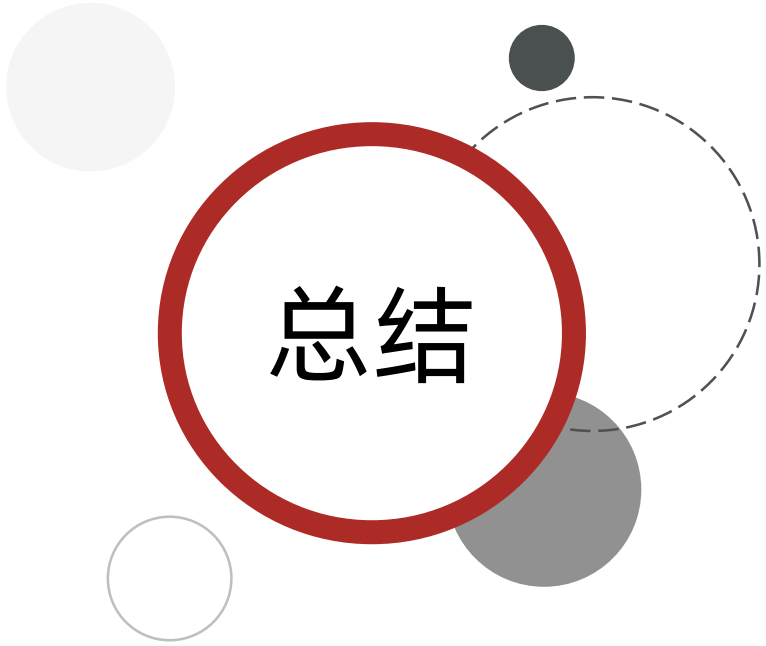
- **概念：**使用 const 声明的变量称为“常量”。
- **使用场景：**当某个变量永远**不会改变**的时候，就可以使用 const 来声明，而不是let。
- **命名规范：**和变量一致
- **常量使用：**

```
// 声明一个常量
const G = 9.8
//输出这个常量
console.log(G)
```

```
// 声明一个常量
const G = 9.8
// 修改常量里面的值
G = 9.9
//输出这个常量
console.log(G)
```

- **注意：** 常量不允许重新赋值,声明的时候必须赋值（初始化）
- **小技巧：**不需要重新赋值的数据使用const






let — 现在实际开发变量声明方式。

var — 以前的声明变量的方式，会有很多问题。

const — 类似于 let，但是变量的值无法被修改。

# 目录

Contents

04

# 数据类型

- 数据类型☆
- 检测数据类型

# 4. 数据类型

**目标：能说出JS中基本数据类型有哪些**

计算机世界中的万事万物都是数据。

计算机程序可以处理大量的数据，为什么要给数据分类？

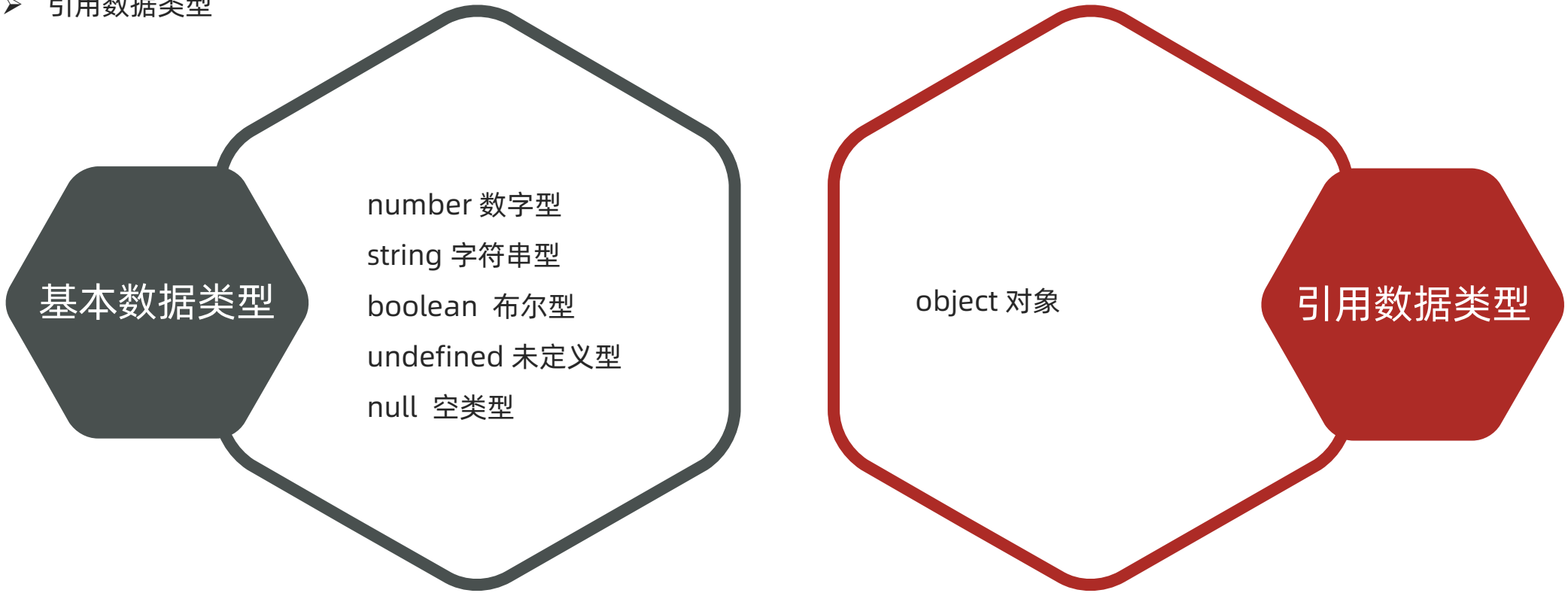
- 1. 更加充分和高效的利用内存
- 2. 也更加方便程序员的使用数据

比如：


## 4. 数据类型

**JS 数据类型整体分为两大类：**

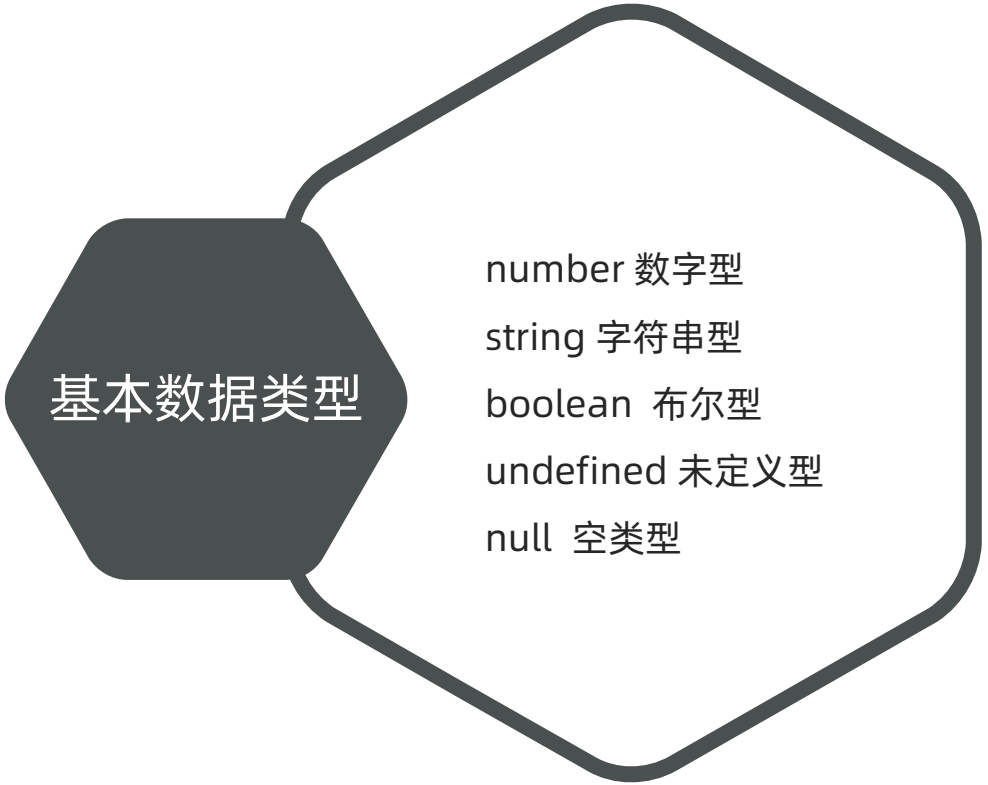
- 基本数据类型
- 引用数据类型



## 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



## 4.1 数据类型 - 数字类型（Number）

即我们数学中学习到的数字，可以是整数、小数、正数、负数。

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

JavaScript 中的正数、负数、小数等 统一称为 数字类型。

**注意事项**

JS 是弱数据类型，变量到底属于那种类型，只有赋值之后，我们才能确认

Java是强数据类型  例如 int a = 3  必须是整数

## 4.1 数据类型 - 数字类型（Number）

数字可以有很多操作，比如，乘法 * 、除法 / 、加法 + 、减法 - 等等，所以经常和算术运算符一起。

数学运算符也叫**算术运算符**，主要包括加、减、乘、除、取余（求模）。

- +：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

## 目标：能说出JavaScript算术运算符执行的优先级顺序

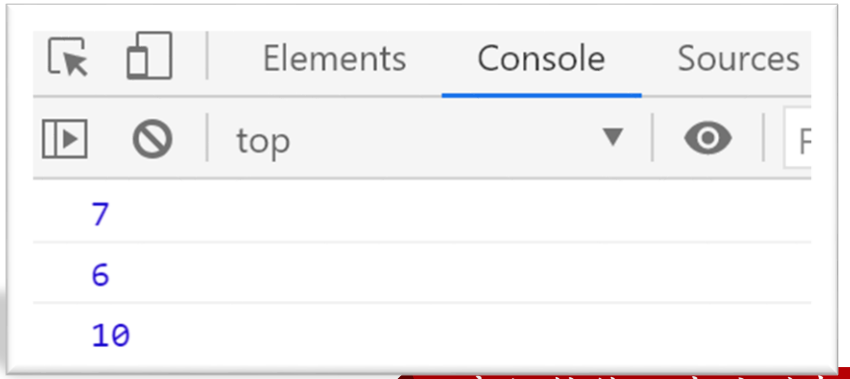
同时使用多个运算符编写程序时，会按着某种顺序先后执行，我们称为优先级。

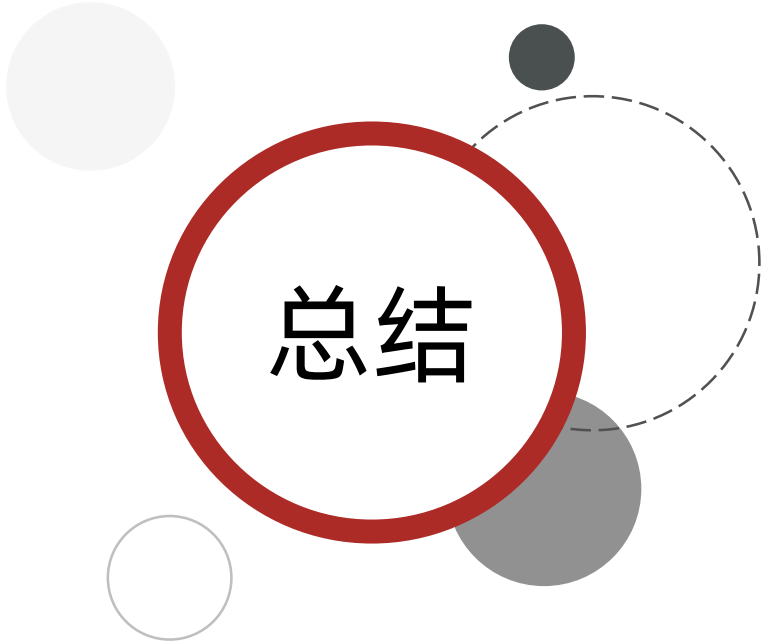
JavaScript中 优先级越高越先被执行，优先级相同时以书从左向右执行。

- 乘、除、取余优先级相同
- 加、减优先级相同
- 乘、除、取余优先级大于加、减
- 使用 () 可以提升优先级
- 总结： 先乘除后加减，有括号先算括号里面的~~~

提问：

```
console.log(1 + 2 * 3)
console.log(10 - 8 / 2)
console.log(2 % 5 + 4 * 2)
```






1. 算术运算符有那几个常见的？
   - \+ - * / %
2. 算术运算符优先级怎么记忆？
   - 先乘除取余，后加减，有小括号先算小括号里面的
3. 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

案例

## 计算圆的面积

需求：对话框中输入圆的半径，算出圆的面积并显示到页面

分析：

①：面积的数学公式： $\pi*r^2$

②：转换为JavaScript写法： 变量 * r * r

## 4.1 数据类型 - 数字类型（Number）

NaN 代表一个计算错误。它是一个不正确的或者一个未定义的数学操作所得到的结果

```
console.log('老师' - 2)  // NaN
```

NaN 是粘性的。任何对 NaN 的操作都会返回 NaN

```
console.log(NaN + 2) // NaN
```

## 4.1 数据类型 - 字符串类型（string）

通过单引号（''）、双引号（""）或反引号(`)包裹的数据都叫字符串，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' // 使用单引号
let gender = "男"  // 使用双引号
let goods = `小米`  // 使用反引号
let tel = '13681113456'  // 看上去是数字，但是引号包裹了就是字符串
let str = '' // 这种情况叫空字符串
```

**注意事项：**

1. 无论单引号或是双引号必须成对使用
2. 单引号/双引号可以互相嵌套，但是不以自已嵌套自已（口诀：外双内单，或者外单内双）
3. 必要时可以使用转义符 \，输出单引号或双引号

## 4.1 数据类型 - 字符串类型（string）

字符串拼接：

**场景**： + 运算符 可以实现字符串的拼接。

口诀：数字相加，字符相连

```
document.write('我叫' + '刘德华')  // 我叫刘德华
let uname = '刘德华'
let song = '忘情水'
document.write(uname + song) // 刘德华忘情水
```

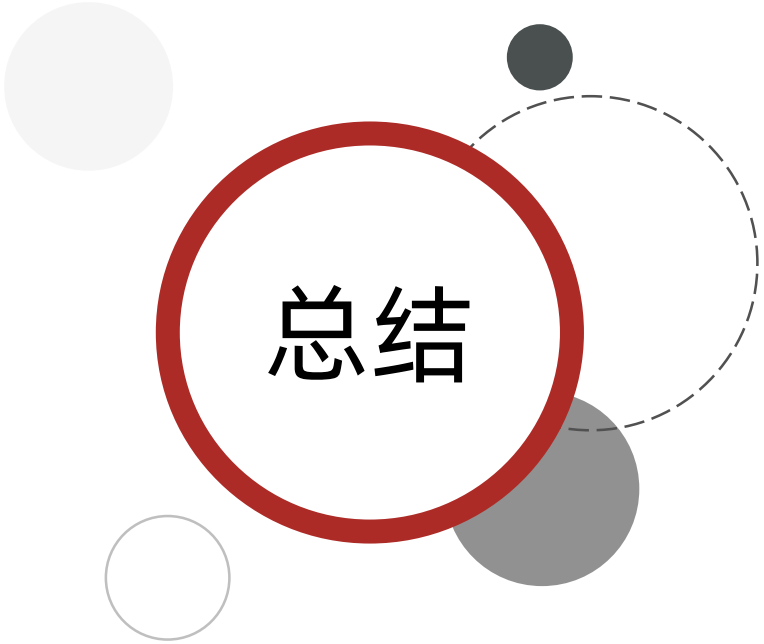
## 模板字符串

- **使用场景**
  - 拼接字符串和变量
  - 在没有它之前，要拼接变量比较麻烦

```
document.write('大家好，我叫' + name + '，今年' + age + '岁')
```

- **语法**
  - ``(反引号)
  - 在英文输入模式下按键盘的tab键上方那个键（1左边那个键）
  - 内容拼接变量时，用 ${ } 包住变量

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




1. JavaScript中什么样数据我们知道是字符串类型？
   - 只要用 单引号、双引号、反引号包含起来的就是字符串类型
2. 字符串拼接比较麻烦，我们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
3. 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}

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

案例

# 页面输出用户信息案例

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

## 4.1 数据类型 - 布尔类型（boolean）

表示肯定或否定时在计算机中对应的是布尔类型数据。

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)
```

## 4.1 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。

## 4.1 数据类型 - null（空类型）

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
```

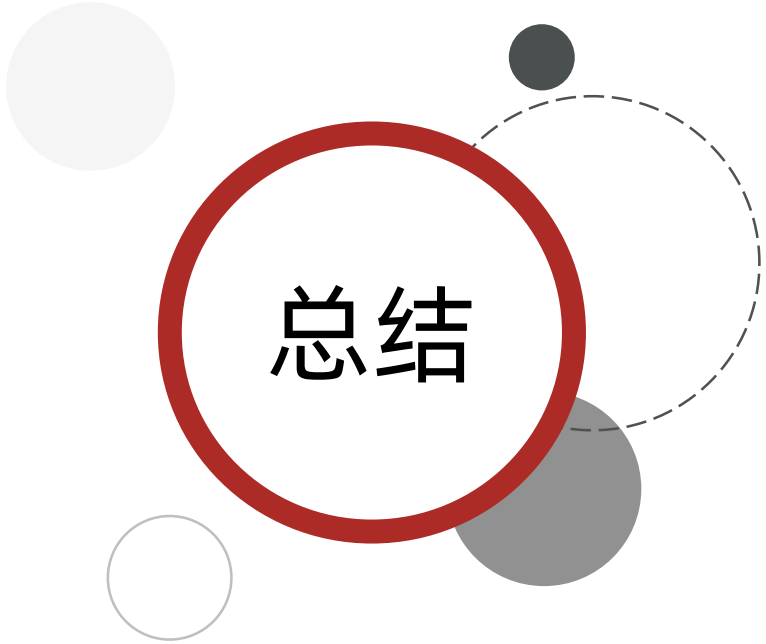
**null 和 undefined 区别：**

- undefined 表示没有赋值
- null 表示赋值了，但是内容为空

**null 开发中的使用场景：**

官方解释：把 null 作为尚未创建的对象

大白话： 将来有个变量里面存放的是一个对象，但是对象还没创建好，可以先给个null




1. 布尔数据类型有几个值？
   - true 和 false
2. 什么时候出现未定义数据类型？以后开发场景是？
   - 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？ 开发场景是？
   - 空类型
   - 如果一个变量里面确定存放的是对象，如果还没准备好对象，可以放个null

04

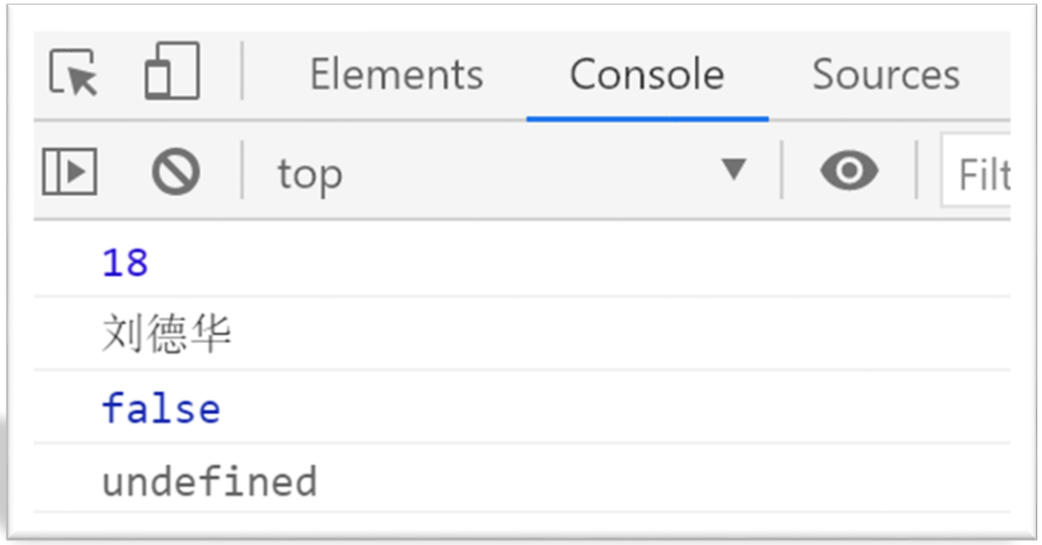
# 数据类型

- 数据类型☆
- 检测数据类型

## 4.2 控制台输出语句和检测数据类型

- **控制台输出语句：**

```
let age = 18
let uname = '刘德华'
let flag = false
let buy
console.log(age)
console.log(uname)
console.log(flag)
console.log(buy)
```




- 控制台语句经常用于测试结果来使用。
- 可以看出数字型和布尔型颜色为蓝色，字符串和undefined颜色为灰色

## 3.2 控制台输出语句和检测数据类型

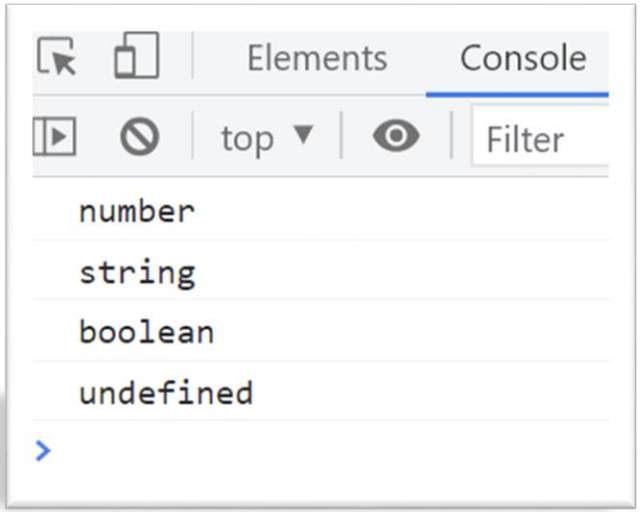
### 2. 通过 typeof 关键字检测数据类型

typeof 运算符可以返回被检测的数据类型。它支持两种语法形式：

1. 作为运算符： typeof x （常用的写法）
2. 函数形式： typeof(x)

换言之，有括号和没有括号，得到的结果是一样的，所以我们直接使用运算符的写法。

```
let age = 18
let uname = '刘德华'
let flag = false
let buy
console.log(typeof age)
console.log(typeof uname)
console.log(typeof flag)
console.log(typeof buy)
```



# 目录

Contents


# 类型转换

- 为什么要类型转换
- 隐式转换
- 显式转换

## 5.1 为什么需要类型转换

JavaScript是弱数据类型： JavaScript也不知道变量到底属于那种数据类型，只有赋值了才清楚。

坑： 使用表单、prompt 获取过来的数据默认是字符串类型的，此时就不能直接简单的进行加法运算。

```
console.log('10000' + '2000')  // 输出结果  100002000
```

此时需要转换变量的数据类型。

通俗来说，就是把一种数据类型的变量转换成我们需要的数据类型。

04

# 类型转换

- 为什么要类型转换
- 隐式转换
- 显式转换

## 5.2 隐式转换

某些运算符被执行时，系统内部自动将数据类型进行转换，这种转换称为隐式转换。

**规则：**

- + 号两边只要有一个是字符串，都会把另外一个转成字符串
- 除了+以外的算术运算符 比如 - * / 等都会把数据转成数字类型

**缺点：**

- 转换类型不明确，靠经验才能总结

**小技巧：**

- +号作为正号解析可以转换成数字型
- 任何数据和字符串相加结果都是字符串

```
<script>
    console.log(11 + 11)
    console.log('11' + 11)
    console.log(11 - 11)
    console.log('11' - 11)
    console.log(1 * 1)
    console.log('1' * 1)
    console.log(typeof '123')
    console.log(typeof +'123')
    console.log(+'11' + 11)
</script>
```

```
22
1111
0
0
1
1
string
number
22
```

## 5.2 显式转换

编写程序时过度依靠系统内部的隐式转换是不严禁的，因为隐式转换规律并不清晰，大多是靠经验总结的规律。

为了避免因隐式转换带来的问题，通常根逻辑需要对数据进行显示转换。

**概念：**

自己写代码告诉系统该转成什么类型

**转换为数字型**

- Number(数据)
  - ✓ 转成数字类型
  - ✓ 如果字符串内容里有非数字，转换失败时结果为 NaN（Not a Number）即不是一个数字
  - ✓ NaN也是number类型的数据，代表非数字
- parseInt(数据)
  - 只保留整数
- parseFloat(数据)
  - 可以保留小数

## 5.2 显式转换

编写程序时过度依靠系统内部的隐式转换是不严禁的，因为隐式转换规律并不清晰，大多是靠经验总结的规律。

为了避免因隐式转换带来的问题，通常根逻辑需要对数据进行显示转换。

**概念：**

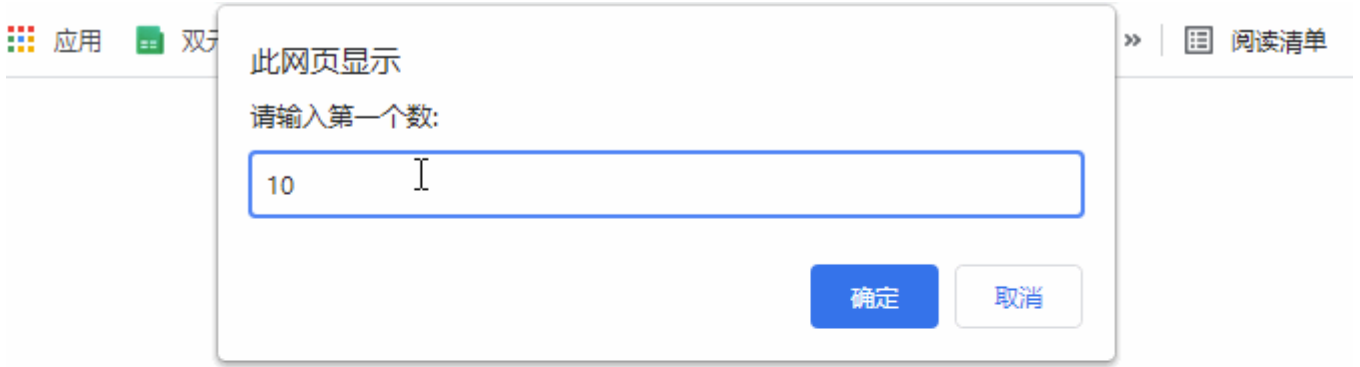
自己写代码告诉系统该转成什么类型

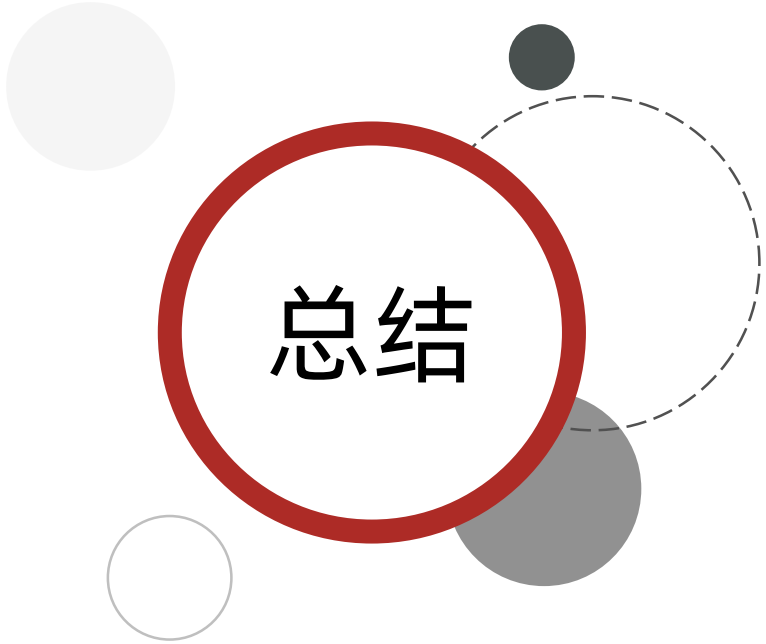
**转换为字符型：**

- String(数据)
- 变量.toString(进制)

练习

- **输入2个数，计算两者的和，打印到页面中**






## 1. 类型转换的概念

一种数据类型转成别的类型， JavaScript是弱数据类型，很多情况计算的时候，需要转换数据类型

## 2. 隐式转换

- 系统自动做转换

## 3. 显式转换

自己写代码告诉系统转成什么类型

- Number

  字符串内容里有非数字得到NaN

- String

# 目录

Contents

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

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：需要输入3个数据，所以需要3个变量来存储　price　num　address

②：需要计算总的价格 total

③：页面打印生成表格，里面填充数据即可

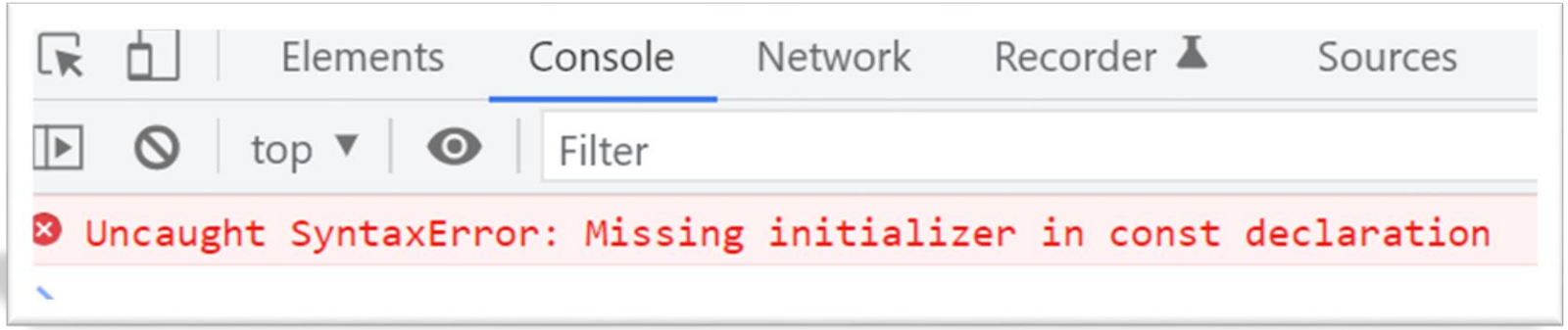
④：记得最好使用模板字符串

**订单确认**

| 商品名称 | 商品价格 | 商品数量 | 总价 | 收货地址 |
| --- | --- | --- | --- | --- |
| 小米手机青春PLUS | 1999元 | 2 | 3998元 | 武汉黑马程序员pink老师收 |

## 六. 常见错误

- 下面可能出现的原因是什么？





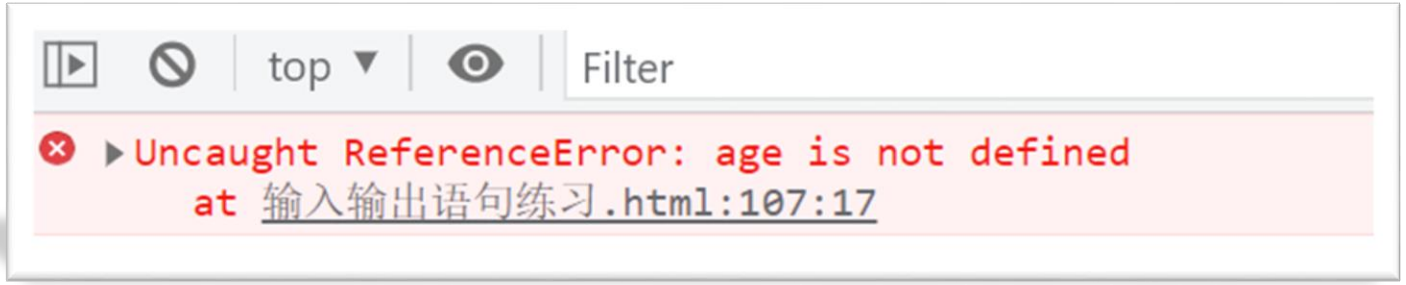
```
const age
console.log(age)  // 错误 const 必须有赋值操作
```

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？



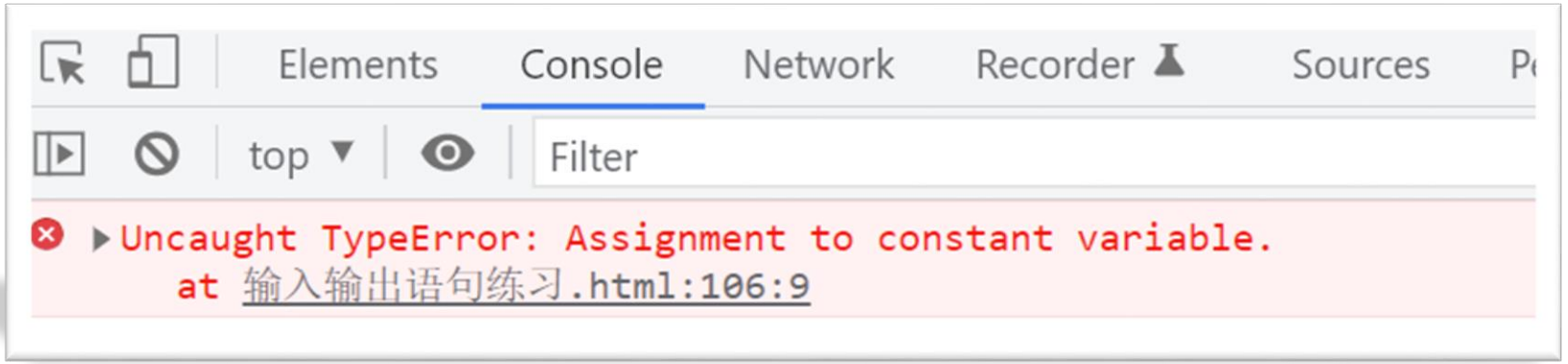
分析：

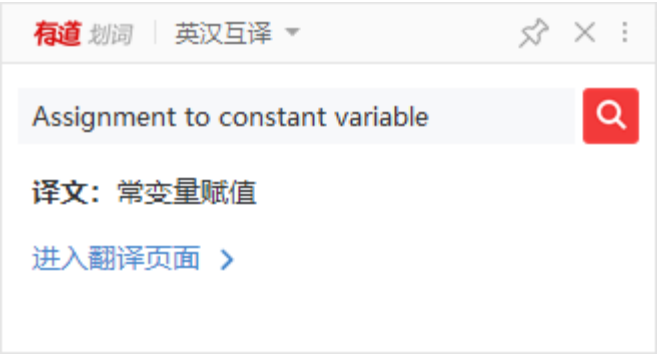
- 提示 “age” 已经声明
- 很大概率是因为重复声明了一个变量。
- 注意let 或者const 不允许多次声明同一个变量

```
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
```

## 六. 常见错误

- 下面可能出现什么问题？如何解决？

```
let num1 = prompt('请输入第一个数值:')
let num2 = prompt('请输入第二个数值:')
alert(`两者相加的结果是: ${num1 + num2}`)
```

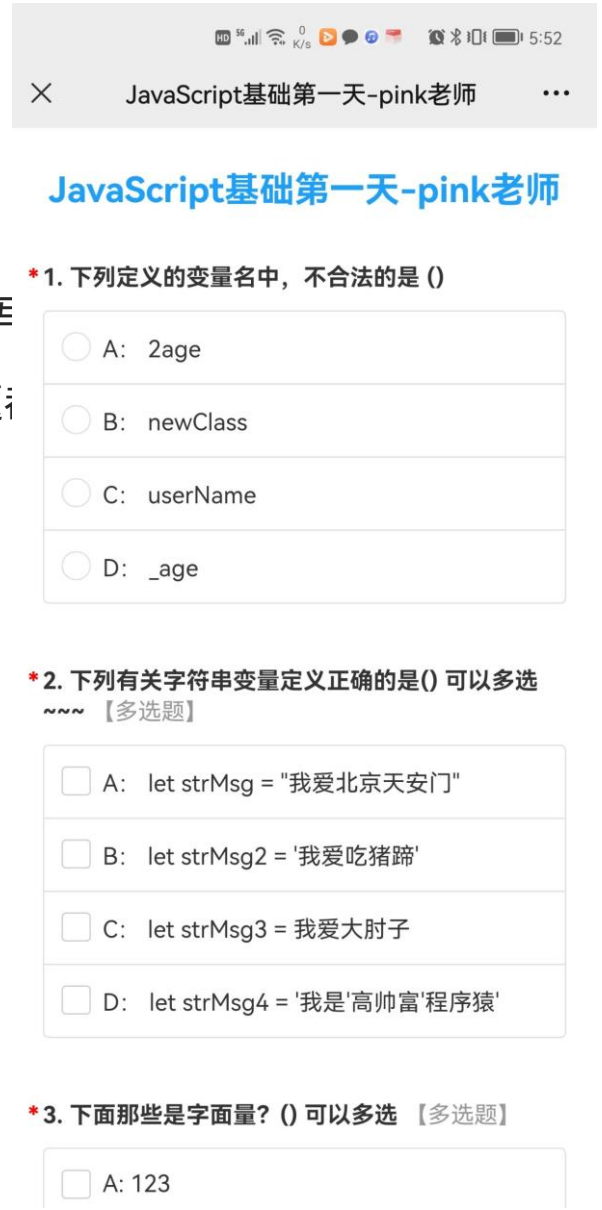
分析：

- 因为prompt 获取过来的是字符型，所以会出现字符相加的问题
- prompt 如果出现相加 记得要转为数字型，可以 利用 + 最简单

```
let num1 = +prompt('请输入第一个数值:')
let num2 = +prompt('请输入第二个数值:')
alert(`两者相加的结果是: ${num1 + num2}`)
```

## 七. 今日复习路线

- 晚自习回来每个同学先必须xmind梳理今日知识点 (md 笔记也行)
- 梳理完毕再次写2遍今日综合案例（订单信息案例案例） 记得先写伪代码思路，然后里面填写
- 晚上8点是做测试题（测试时间最多30分钟） 注意，必须每个同学都完全做对。保证18个题
- 做完考题的同学开始：独立书写今日作业
- 手机扫码：　　电脑端： https://ks.wjx.top/vj/h8kAn6p.aspx




传智教育旗下高端IT教育品牌