**题目1：声明多个变量分别保存一个人的姓名，性别，电话，家庭地址。并打印出来。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const name = "张建安";  const gender = "男";  const phone = "13033458033";  const address = "山西省大同市";  console.log(`姓名：${name}，性别：${gender}，电话：${phone}，地址：${address}`); |

**题目2：声明变量分别保存语文、数学、总成绩，其中总成绩为空；把语文和数学相加的和赋给总成绩。最后打印3个变量。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const chineseScore = 100;  const mathScore = 100;  let totalScore;  totalScore = chineseScore + mathScore;  console.log(`语文成绩：${chineseScore}，数学成绩：${mathScore}，总成绩：${totalScore}`); |

**题目3：声明变量保存圆周率，声明变量保存圆的半径，计算周长和面积，声明变量分别保存圆的面积和周长。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const pi = 3.14159;   // 圆周率  const radius = 5;     // 半径  const circumference = 2 \* pi \* radius;   // 周长  const area = pi \* radius \*\* 2;           // 面积  console.log(`圆周长：${circumference}，圆面积：${area}`); |

**题目4：交换两个变量的值 a=1 b=2，得到 a=2 b=1。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  let a = 1;  let b = 2;  [a, b] = [b, a];  console.log(`a=${a} b=${b}`); |

**题目5： 假设从数据库获取到了商品的名称和价格，拼接打印出’商品名称：xxx 价格：xxx’。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const shopname = "iPhone";  const price = 6000;  console.log(`商品名称：${shopname} 价格：${price}`); |

**题目6：声明变量保存输入的用户名和密码，如果用户名为’root’并且密码为’123abc’，则打印true，否则打印false。 (并且符号： &&)。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const userName = "root";  const password = "123abc";  console.log(userName === "root" && password === "123abc"); // 输出true |

**题目7：JavaScript的数据类型有哪几种，请列举出来。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  6种  基本数据类型：Undefined, Null, Boolean, Number, String  对象类型：Object |

**题目8：简述undefined与null的关系。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  undefined表示变量未定义，null表示变量定义了，但值为空。  undefined和null在类型转换时，都转换为undefined。  两者都是可以赋值给其他变量的，但undefined变量不能赋值给null变量。 |

**题目9：基本数据类型和引用数据类型在内存中如何存储。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  基本数据类型存储在栈内存中，引用数据类型存储在堆内存中。 |

**题目10：简述深拷贝与浅拷贝。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  浅拷贝：拷贝对象时，只复制对象的引用，而不是对象本身。  深拷贝：拷贝对象时，复制对象本身及其属性。 |

**题目1：分析以下程序的运行结果。**

var n = 7;

var n1 = n--;

var n2 = --n;

console.log(n1 + n2);

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  var n = 7; 初始化变量n为7。  var n1 = n--; 这里使用了后递减操作符--。它首先返回变量n的当前值给n1，然后将n的值减1。n1被赋值为7，n变为6。  var n2 = --n; 这里使用了前递减操作符--。它首先将n的值减1，然后返回新的值给n2。n变为5，n2被赋值为5。 |

**题目2：声明变量保存一个人的年龄，如果满18岁打印’成年人’，使用三目运算符或短路逻辑。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const age = 17;  console.log(age >= 18 ? '成年人' : '未成年人'); |

**题目3：声明变量保存年份，如果是闰年打印’闰年’，否则打印’平年’（闰年的判断：能被4整除，并且不能被100整除，或者能被400整除）。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const year = 2024;  console.log(year % 4 === 0 && year % 100 !== 0 || year % 400 === 0 ? '闰年' : '平年'); |

**题目4：声明变量保存用户输入的用户名和密码，如果用户名为root并且密码为123456，打印‘登录成功’，否则‘登录失败’。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const userName = 'root';  const password = '123456';  console.log(userName === 'root' && password === '123456' ? '登录成功' : '登录失败'); |

**题目5： 声明变量保存性别的值(1/0)，如果是1打印男，否则打印女（分别使用if-else 和 三目运算符）。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const gender = 1;  console.log(gender === 1 ? '男' : '女');  if (gender === 1) {      console.log('男');  } else {      console.log('女');      } |

**题目6：声明变量保存订单的状态码，根据订单的状态码打印对应的汉字状态。 (1-等待付款 2-等待发货 3-运输中 4-已签收 5-已取消 其它-无法追踪)。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const orderStatus = 2;      switch (orderStatus) {          case 1:              console.log('等待付款');              break;          case 2:              console.log('等待发货');              break;          case 3:              console.log('运输中');              break;          case 4:              console.log('已签收');              break;              case 5:              console.log('已取消');              break;              default:              console.log('无法追踪');              break;              } |

**题目7：声明变量保存价格，假设打5折，计算结果是多少并打印出来。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述    const price = 100;              console.log(price - price \* 0.5); |

**题目8：声明变量保存一个人的成绩，根据分数做出评判，成绩范围为0~100，超出此范围输出非法的成绩（90~ 优秀，80~90以下 良好，70~80以下 中等，60~70以下 及格，60以下 不及格）。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述   const score = 89;              switch (true) {                  case score > 100:                      console.log('非法的成绩');                      break;                  case score >= 90:                      console.log('优秀');                      break;                  case score >= 80:                      console.log('良好');                      break;                  case score >= 70:                      console.log('中等');                      break;                  case score >= 60:                      console.log('及格');                      break;                      default:                      console.log('不及格');                      break;                      } |

**题目9：使用冒牌排序算法，将数组[74,100,27,4,58,93,61]内的元素从小到大排列。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述   const arr = [74, 100, 27, 4, 58, 93, 61];                      for (let i = 0; i < arr.length - 1; i++) {                          for (let j = i + 1; j < arr.length; j++) {                              if (arr[i] > arr[j]) {                                  [arr[i], arr[j]] = [arr[j], arr[i]];                              }                          }                      }                      console.log(arr); |

**题目1：假设张三有10000元，银行的利率是4%，5年后张三一共能拿到多少钱（一年后的钱 = 前一年的本金 + 利息）。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const calculateTotalAmountHelper = (initialAmount, interestRate, years) => {      if (years === 0) {          return initialAmount;      }      return calculateTotalAmountHelper(initialAmount \* (1 + interestRate), interestRate, years - 1);  };  console.log(calculateTotalAmountHelper(10000, 0.04, 5)); |

**题目2：计算1~100之间所有整数的和，一旦当总和超过4000，结束循环，并打印当前的和。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const calculateSum = (start, end) => {      let sum = 0;      for (let i = start; i <= end; i++) {          sum += i;          if (sum > 4000) {              console.log('当前和:', sum);              break;          }      }  };  calculateSum(1, 100); |

**题目3：打印2000~2100之间的前10个闰年。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const isLeapYear = year => {      return (year % 4 === 0 && year % 100 !== 0) || year % 400 === 0;  };  for (let i = 2000; i <= 2100; i++) {      if (isLeapYear(i)) {          console.log(i);      }  } |

**题目4：打印九九乘法表。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  for (let i = 1; i <= 9; i++) {      for (let j = 1; j <= i; j++) {          console.log(`${i} \* ${j} = ${i \* j}`);      }  } |

**题目5：使用\*号输出，打印出直角三角形。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  for (let i = 1; i <= 9; i++) {      console.log('\*'.repeat(i));  } |

**题目6：鸡兔同笼一共50只，脚160，求鸡多少只，兔子多少只？**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  const calculateChickensAndRabbits = (total, feet) => {      const chickens = Math.floor(feet / 2);      const rabbits = Math.floor((total - chickens \* 2) / 2);      console.log(`鸡: ${chickens}, 兔子: ${rabbits}`);      return [chickens, rabbits];  };  calculateChickensAndRabbits(50, 160); |

**题目7：简述目前所知的对象的创建方式，并分别列举出对象中属性的增删改查的方式。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  字面量方式：直接使用花括号 {} 创建对象。  构造函数：使用 new 关键字和构造函数创建对象。  Object.create()：使用 Object.create() 方法创建新对象。  类（ES6及以后）：使用 class 关键字定义类，然后通过 new 关键字创建实例。    // 字面量方式  let obj1 = { name: "Alice", age: 25 };  // 构造函数  function Person(name, age) {      this.name = name;      this.age = age;  }  let obj2 = new Person("Bob", 30);  // Object.create()  let obj3 = Object.create({ name: "Charlie", age: 35 });  // 类  class Student {      constructor(name, age) {          this.name = name;          this.age = age;      }  }  let obj4 = new Student("Dave", 28);  // 增加属性  obj1.occupation = "Engineer";  // 修改属性  obj2.name = "Charlie";  // 查询属性  console.log(obj3.name); // 输出: Charlie  // 删除属性  delete obj4.name; |

**题目8：请使用图例或文字说明Object原型属性及方法。**

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  toString()：返回对象的字符串表示。  valueOf()：返回对象的数值表示。  hasOwnProperty(prop)：检查对象自身是否具有指定的属性。  \*sPrototypeOf(obj)：检查当前对象是否是另一个对象的原型。  propertyIsEnumerable(prop)：检查属性是否可枚举。  toLocaleString()：返回对象的本地化字符串表示。  toJSON()：返回对象的JSON字符串表示。  constructor：指向创建当前对象的构造函数。  Eg：  let obj = {};  console.log(obj.toString()); // [object Object]  console.log(obj.valueOf()); // [object Object]  console.log(obj.hasOwnProperty('toString')); // true  console.log(Object.prototype.isPrototypeOf(obj)); // true  console.log(Object.prototype.propertyIsEnumerable('toString')); // true  console.log(obj.toLocaleString()); // [object Object]  console.log(JSON.stringify(obj)); // '{}'  console.log(obj.constructor); // [Object] |

1.代码题:为页面中的按钮绑定事件，实现点击按钮改变页面中某个元素的背景颜色以及其中文本内容的功能。

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <title>AJAX作业</title>  <style>  #myElement {  padding: 20px;  background-color: blue;  text-align: center;  }  </style>  </head>  <body>  <div id="AAA">点击按钮更改我的背景和内容</div>  <button id="Button">点击我</button>  <script>  // 获取按钮和元素  var button = document.getElementById('Button');  var element = document.getElementById('AAA');  // 绑定点击事件  button.addEventListener('click', function() {  // 更改背景颜色  element.style.backgroundColor = 'lightcoral';  // 更改文本内容  element.textContent = '背景和内容已改变！';  });  </script>  </body>  </html> |

2.简述XMLHttpRequest的使用方法及步骤。

**答案：**

|  |
| --- |
| 将具体答案写入方框中，注意代码中必须添加注释描述  1. 创建 XMLHttpRequest 对象  var xhr = new XMLHttpRequest();  2. 配置请求  使用 open() 方法配置请求的类型、URL 和是否异步：  xhr.open('GET', 'https://api.example.com/data', true);  第一个参数是请求方法（如 'GET' 或 'POST'）。  第二个参数是请求的 URL。  第三个参数是布尔值，表示是否异步（ true 是异步， false 是同步）。  3. 处理响应  设置onreadystatechange事件处理函数，以便在响应状态改变时处理数据：  xhr.onreadystatechange = function() {  if (xhr.readyState === 4 && xhr.status === 200) {  // 请求完成且成功  var response = xhr.responseText;  console.log(response);  }  };  xhr.readyState表示请求的状态，4 表示请求已完成。  xhr.status表示 HTTP 响应状态码，200 表示请求成功。  4. 发送请求  使用send()方法发送请求：  xhr.send();  对于 POST 请求，你可能需要设置请求头并发送数据：  xhr.open('POST', 'https://api.example.com/submit', true);  xhr.setRequestHeader('Content-Type', 'application/json');  xhr.send(JSON.stringify({ key: 'value' }));  完整代码如下：  var xhr = new XMLHttpRequest();  xhr.open('GET', 'https://api.example.com/data', true);  xhr.onreadystatechange = function() {  if (xhr.readyState === 4 && xhr.status === 200) {  var response = xhr.responseText;  console.log(response);  }  };  xhr.send();  主要属性和方法  属性：  xhr.readyState ：请求的当前状态（0-4）。  xhr.status ：HTTP 响应状态码。  xhr.responseText ：响应的文本内容。  xhr.responseXML ：响应的 XML 文档（如果响应类型为 XML）。  方法：  xhr.open(method, url, async) ：配置请求。  xhr.send(body) ：发送请求（ body 参数用于 POST 请求）。  xhr.setRequestHeader(header, value) ：设置请求头。 |