**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] |