师资培训课件（函数封装）

> 难度： ⭐⭐ 到 ⭐⭐⭐⭐⭐

1. 十三届省赛谁最长

```
/**
 * 封装函数，传入任意数量的数组，返回长度最大的数组集合（长度相同且为最大的数组存入集合，如果所有数组长度相同，则返回空数组），并将它们分别按升序排序
 */
const getMaxArrays = (...arrays) => {
```

```
  if (!arrays || arrays.length <= 0) {
    return [];
  }
  let maxArray = [arrays[0]];
  let j;
  for (let i = 1; i < arrays.length; i++) {
    if (!Array.isArray(arrays[i])) {
      return [];
    }
    // 传入数组与返回数组最后一个比较
    j = maxArray.length - 1;
    // 如果传入数组不是当前最大，则继续下一轮比较
    if (arrays[i].length < maxArray[j].length) {
      continue;
    } else {
      // 如果是当前最大，则清空返回数组，并存入
      if (arrays[i].length > maxArray[j].length) {
        maxArray = [];
        maxArray.push(arrays[i]);
      }
      // 如果与当前最大长度相同，则存入
      else {
        maxArray.push(arrays[i]);
      }
    }
  }

  if (maxArray.length == arrays.length) {
    maxArray = [];
  }
  return maxArray; // 正序输出
};
module.exports = getMaxArrays; //请勿删除
```

2.十三届省赛寻找小狼人

```
Array.prototype.myarray = function (cb) {
  // TODO 带补充代码
  var filteredArray = [];
  for (var i = 0; i < this.length; i++) {
    if (cb(this[i])) {
      filteredArray.push(this[i]);
    }
  }
  return filteredArray;
};
```

3. 十三届国赛分一分

```
/**
 * @param {Object} oldArr
 * @param {Object} num
 * */
const splitArray = (oldArr, num) => {
  // 处理待分割数据为空数组或非数组的情况
  if (!oldArr || oldArr.length == 0) {
    return [];
  }
  // 处理待分割数据为正常数组的情况
  let sortedArr = oldArr.sort((a, b) => a - b);
  // 把一个数组分割成每 num 个 1 组
  let arr = [];
  let len = sortedArr.length;
  for (let i = 0; i < len; i += num) {
    arr.push(sortedArr.slice(i, i + num));
  }
  return arr;
};
module.exports = splitArray; // 检测需要，请勿删除
```

4.十三届国赛猜硬币

```
/**
 * @param {*} input_values input 框中输入的值
 * @param {*} res 输入的值组成的一个新的数组
 * @returns
 */
// 将输入的值中的 `1-9` 数字组成的一个新的数组

function findNum(input_values) {
  let input_values_arr = input_values.split("");
  let res = [];
  // 取出只是数字的部分
  for (let index = 0; index < input_values_arr.length; index++) {
    let input_value = input_values_arr[index];
    if (!isNaN(input_value) && input_value != 0) {
      res.push(Number(input_value));
    }
  }
  res = Array.from(new Set(res));
  return res;
}

//随机生成硬币位置  随机三个 1-9中 三个不重复的随机数 并放入数组中并返回这个数组
let randomCoin = () => {
  let randomNumArr = [];
  while (randomNumArr.length < 3) {
    let num = Math.floor(Math.random() * 9) + 1;
    if (randomNumArr.indexOf(num) == -1) {
      randomNumArr.push(num);
```

```
    }
  }
  return randomNumArr;
};

// 请勿删除和修改下面代码
try {
  module.exports = { randomCoin, findNum };
} catch (e) {}
```

5.十三届国赛用什么来做计算

```
// 结果显示框
let result = document.querySelector("#result");
// 计算式子显示框
let formula = document.querySelector("#formula");
// 所有按钮
buttons = document.querySelectorAll(".calc-button");
// 初始化式子为空
let formulaValue = "";
// 初始化结果值为空
let resultValue = "";
// 计算符号
let mark = "";
// 计数，用于判断等号按钮是否被点击
let count = 0;
// 监听按钮的点击事件
for (item of buttons) {
  item.addEventListener("click", (e) => {
    // 获取按钮中的文本
    buttonText = e.target.innerText;
    // 判断是否为乘号
    if (buttonText == "x") {
      buttonText = "*";
      mark = "x";
      if (count == 0) {
        formulaValue += mark;
        formula.value = formulaValue;
        resultValue += buttonText;
      } else {
        formulaValue = resultValue;
        formulaValue += mark;
        formula.value = formulaValue;
        resultValue += buttonText;
      }
    }
    // 判断是否为清空按钮
    else if (buttonText == "AC") {
      resultValue = "";
```

```
    formulaValue = "";
    formula.value = formulaValue;
    result.value = resultValue;
  }
  // 判断是否为除号
  else if (buttonText == "÷") {
    buttonText = "/";
    mark = "÷";
    if (count == 0) {
      formulaValue += mark;
      formula.value = formulaValue;
      resultValue += buttonText;
    } else {
      formulaValue = resultValue;
      formulaValue += mark;
      formula.value = formulaValue;
      resultValue += buttonText;
    }
  } else if (buttonText == "(") {
    buttonText = "(";
    mark = "(";
    formulaValue += mark;
    formula.value = formulaValue;
    resultValue += buttonText;
  } else if (buttonText == ")") {
    buttonText = ")";
    mark = ")";
    formulaValue += mark;
    formula.value = formulaValue;
    resultValue += buttonText;
  }
  // 判断是否为等号
  else if (buttonText == "=") {
    let value = eval(resultValue);
    resultValue = value;
    result.value = value;
    count++;
  }
  // 判断是否为开方号
  else if (buttonText == "√") {
    resultValue = Math.sqrt(formula.value, 2);
    result.value = resultValue;
    formulaValue = resultValue;
  }
  // 判断是否为加号
  else if (buttonText == "+") {
    buttonText = "+";
    mark = "+";
    if (count == 0) {
      formulaValue += mark;
      formula.value = formulaValue;
      resultValue += buttonText;
    } else {
      formulaValue = resultValue;
      formulaValue += mark;
      formula.value = formulaValue;
      resultValue += buttonText;
```

```
      }
    }
    // 判断是否为减号
    else if (buttonText == "-") {
      buttonText = "-";
      mark = "-";
      if (count == 0) {
        formulaValue += mark;
        formula.value = formulaValue;
        resultValue += buttonText;
      } else {
        formulaValue = resultValue;
        formulaValue += mark;
        formula.value = formulaValue;
        resultValue += buttonText;
      }
    } else {
      formulaValue += buttonText;
      formula.value = formulaValue;
      resultValue += buttonText;
    }
  });
}
```

6.十三届国赛JSON 生成器

```
// 随机生成布尔值
function bool() {
  return !!Math.floor(2 * Math.random());
}
// 随机取 n 到 m 之间的整数
function integer(n, m) {
  return Math.floor(Math.random() * (m - n + 1)) + n;
}

// 生成对象的逻辑
let genobj = (data) => {
  let obj = {};
  for (let key in data) {
    let fn;
    //1. 如果值是字符串并且包含双大括号 则获取双大括号中间的值
    if (typeof data[key] == "string" && data[key].includes("{{") &&
data[key].includes("}}")) {
      fn = data[key].slice(2, -2);
      if (fn && fn.includes("(") && fn.includes(")")) {
        try {
          // 把大括号中间的字符串作为函数执行
          let fnkey = new Function("return " + fn)();
          obj[key] = fnkey;
        } catch (err) {
          // 如果不是可执行函数则需要返回原值
```

```
          obj[key] = data[key];
        }
      } else {
        obj[key] = data[key];
      }
    } else {
      //2. 其他情况返回原值
      obj[key] = data[key];
    }
  }
  return obj;
};

let generateData = (data) => {
  let result;
  // 判断是数组还是对象
  if (Array.isArray(data)) {
    // 判断数组第一项是否是 repeat
    if (Array.isArray(data) && typeof (data[0]) == "string") {
      // 设置数组和数组长度长度
      let arr = [], arrlength = 0;
      // 获取 repeat 的中的数字
      let num = data[0].slice(9, -3);
      if (num.length == 1) {
        // 获取到的 num 是字符传，需要转化成数字
        arrlength = +num;
      } else {
        // 随机数组长度
        let numarr = num.split(',')
        arrlength = integer(Number(numarr[0]), Number(numarr[1]));
      }
      // 循环数组，生成数据数组中的每一项数据
      for (let i = 0; i < arrlength; i++) {
        let item = genobj(data[1])
        arr.push(item);
      }
      result = arr;

    } else {
      // 如果没有 repeat 则里面数组第一项是对象
      result = [genobj(data[0])];
    }
  } else {
    result = genobj(data);
  }
  return result;
};



try {
  module.exports = { generateData };
} catch (e) {
}
```

7. 十三届国赛商城管理系统

```
/**
 * @param {*} menuList 传入的数据
 * @return {*} menus 转化后的树形结构数据，auths 转化后的权限列表数组
 */
const getMenuListAndAuth = (menuList) => {
  let menus = [];
  let sourceMap = {};
  let auths = [];
  menuList.forEach((m) => {
    m.children = []; // 增加孩子列表
    sourceMap[m.id] = m;
    auths.push(m.auth);
    if (m.parentId === -1) {
      menus.push(m); // 根节点
    } else {
      sourceMap[m.parentId] && sourceMap[m.parentId].children.push(m);
    }
  });
  return { menus, auths }; // 获取菜单数据和权限数据，请勿修改此行代码
};
```

8. 十四届省赛 ISBN 码校验与生成

```
// 将用户输入的带分隔符的isbn字符串转换只有纯数字和大写X字母的字符串
function getNumbers(str) {
  return str.replace(/[^0-9xX]/g, "").toUpperCase();
}

// 验证当前ISBN10字符串是否有效
/*
  有效的ISBN10字符串的前九位可以是任意数字，最后一位校验位的值取决于前九位数字。

  校验位计算方法：用1-9这9个数依次乘以前面的9位数，然后求它们的和除以11的余数。如果余数为10，则校验码用"X"表示。

  以7-5600-3879-4为例，它的前9位数是7、5、6、0、0、3、8、7、9。它的校验码的计算如下：

  1x7+2x5+3x6+4x0+5x0+6x3+7x8+8x7+9x9
  = 7+10+18+0+0+18+56+56+81
  = 246

  246 % 11 = 4

  因此，这个ISBN的校验码就是4。
*/
function validISBN10(isbn) {
  if (isbn.length !== 10) {
```

```
    return false;
  }
  let pre = isbn.slice(0, 9);
  let last = 0;
  for (let i = 0; i < pre.length; i++) {
    last += (i + 1) * parseInt(pre[i]);
  }
  last %= 11;
  if (last === 10) {
    last = "X";
  }
  if (isbn.slice(-1) == last) {
    return true;
  } else {
    return false;
  }
}

// 将用户输入的ISBN-10字符串转化为ISBN-13字符串

/* 转化方法
  1. 将ISBN-10的最后一位校验位去掉，剩下前九个数字。
  2. 在字符串开头增加978三个数字，获得长度为12的数字字符串。
  3. 计算最后一位校验位。

  13位ISBN的校验码计算规则是这样的：前12位数依次乘以1和3，然后求它们的和除以10的余数，最后用10减去这个余数，就得到了校验码。如果余数为0，则校验码为0.

  比如，7-5600-3879-4在13位ISBN中，就是978-7-5600-3879-4。它的校验码计算方法如下：

  9x1+7x3+8x1+7x3+5x1+6x3+0x1+0x3+3x1+8x3+7x1+9x3
  = 9+21+8+21+5+18+0+0+3+24+7+27
  = 143

  143 % 10 = 3
  10 - 3 = 7
*/
function ISBN10To13(isbn) {
  let pre = "978" + isbn.slice(0, 9);
  let last = 0;
  for (let i = 0; i < pre.length; i++) {
    if (i % 2 === 0) {
      last += parseInt(pre[i]);
    } else {
      last += parseInt(pre[i]) * 3;
    }
  }
  pre += last % 10 !== 0 ? 10 - (last % 10) : 0;
  return pre;
}

// 测试用例
console.log(getNumbers("7-5600-3879-4")); // 7560038794
console.log(getNumbers("7 5600 3879 4")); // 7560038794

console.log(validISBN10("7560038794")); // true
console.log(validISBN10("7560038793")); // false
```

```
console.log(validISBN10("756003879")); // false
console.log(validISBN10("756003879004")); // false

console.log(isbn10To13("7560038794")); // 9787560038797
console.log(isbn10To13("3598215088")); // 9783598215087
```

9. 十四届国赛抢红包了

```
function randomAllocation(total, n) {
  var remain = total;
  var ret = []; // 最终返回的数据

  // 分配金额循环 n-1 次
  for (let i = 0; i < n - 1; i++) {
    // m 为每次分配的红包金额，取 0.1 和 remain 中间的随机数
    let m =
      Math.ceil(Math.random() * 100 * (remain - (n - (i + 1)) * 0.01)) /
100;
    ret.push(m);
    remain -= m; // 剩余量每次减去 M
  }
  ret.push(Number(remain.toFixed(2)));
  return ret;
}
```

10. 十四届国赛版本比较器

```
function compareVersion(version1, version2) {
  const regex = /^\d+(\.\d+){0,2}$/; // 匹配 1~3 个数字，中间用 . 分隔的正则表达式
  if (!regex.test(version1) || !regex.test(version2)) {
    return "error"; // 如果版本号格式不正确，则返回 "error"
  }

  const v1Arr = version1.split(".").map(Number); // 将 version1 按 . 分隔后，
将每个部分转换为数字
  const v2Arr = version2.split(".").map(Number); // 将 version2 按 . 分隔后，
将每个部分转换为数字
  const len = Math.max(v1Arr.length, v2Arr.length); // 获取两个版本数组的最大长
度

  for (let i = 0; i < len; i++) {
    const v1 = i < v1Arr.length ? v1Arr[i] : 0; // 如果 version1 的长度大于
i，则获取第 i 个部分，否则为 0
    const v2 = i < v2Arr.length ? v2Arr[i] : 0; // 如果 version2 的长度大于
i，则获取第 i 个部分，否则为 0
```

```
    if (v1 > v2) {
      return 1; // 如果 version1 大于 version2，则返回 1
    } else if (v1 < v2) {
      return -1; // 如果 version1 小于 version2，则返回 -1
    }
  }
  return 0; // 如果两个版本号相等，则返回 0
}
```

高频考点知识点讲解：

1. 递归函数（分数较高的题目中出现，基础较好的学生一定要掌握）

递归函数是指在函数体内调用自身的函数。递归函数通常用于解决可以分解为相同问题的子问题的情况。

```
function isPalindrome(str) {
    if (str.length <= 1) {
        return true;
    }
    if (str[0] !== str[str.length - 1]) {
        return false;
    }
    return isPalindrome(str.slice(1, -1));
}

const text = 'racecar';
console.log(isPalindrome(text)); // 输出 true
```

2. 回调函数 (逻辑较为简单，建议一定要掌握，通常用来处理异步操作)

在一个函数中，传递了函数作为参数，就叫回调函数 。

```
// 模拟异步操作，比如从服务器获取数据
function fetchData(callback) {
    setTimeout(() => {
        const data = "这是从服务器获取的数据";
        callback(data); // 在异步操作完成后调用回调函数，并将数据传递给它
    }, 2000); // 模拟2秒延迟
}

// 回调函数，用于处理从服务器获取的数据
function processData(data) {
    console.log("处理数据:", data);
}
```

```
// 调用 fetchData 函数，并传递回调函数作为参数
fetchData(processData);
```

3. 数组方法(重点：会贯穿在很多题目里面进行考察)

数组方法用于在数组上执行各种操作，如添加、删除、遍历等。

```
const array = [1, 2, 3, 4, 5];
array.splice(2, 1, 10); // 从索引2开始，删除一个元素，并插入10
console.log(array); // 输出 [1, 2, 10, 4, 5]

// splice() 方法可以用于删除数组中的元素而不替换它们。下面是一个示例：
const array = [1, 2, 3, 4, 5];
array.splice(2, 1); // 从索引2处删除一个元素
console.log(array); // 输出 [1, 2, 4, 5]

// 使用数组方法过滤出偶数
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
const evenNumbers = numbers.filter(num => num % 2 === 0);
console.log(evenNumbers); // 输出 [2, 4, 6, 8, 10]

// 使用数组方法求和
const sum = numbers.reduce((acc, curr) => acc + curr, 0);
console.log(sum); // 输出 55
```

4. 字符串方法

字符串方法用于在字符串上执行各种操作，如查找、替换、分割等。

```
// 使用字符串方法查找子串
const str = "Hello, world!";
console.log(str.indexOf("world")); // 输出 7

// 使用字符串方法替换子串
const newStr = str.replace("world", "JavaScript");
console.log(newStr); // 输出 "Hello, JavaScript!"

// 使用字符串方法分割字符串
const words = str.split(", ");
console.log(words); // 输出 ["Hello", "world!"]
```

5. 随机数

随机数用于生成随机值，通常用于模拟真实世界的情况或增加变化性。

```
// 生成随机整数
const randomNumber = Math.floor(Math.random() * 10) + 1; // 生成 1 到 10 之间的随机整数
console.log(randomNumber);

// 生成随机浮点数
const randomFloat = Math.random() * 10; // 生成 0 到 10 之间的随机浮点数
console.log(randomFloat);
```

6. 二维数组操作(一般会在 20 分以上的题目出现，对逻辑要求较高)

二维数组是由数组组成的数组，可以使用双重循环对其进行操作。

```
// 创建一个二维数组
const matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
];

// 访问二维数组元素
console.log(matrix[1][2]); // 输出 6

// 遍历二维数组
for (let i = 0; i < matrix.length; i++) {
    for (let j = 0; j < matrix[i].length; j++) {
        console.log(matrix[i][j]);
    }
}
```

7. 对象的基础操作

以上是关于JavaScript中递归函数、回调函数、数组方法、字符串方法、随机数和二维数组操作的简单示例。

总结： 函数封装占分比重 30 - 70 分。题目数量：2 - 4。高频考点：递归函数 、回调函数、数组方法、字符串方法、随机数、二维数组、复杂数据结构的操作

复杂数据结构： 多层对象数组嵌套 分数较高，难度并不会特别难，操作可能会复杂，比较耗时，比

较需要细心。 基本上属于高分题目必考点。时间足够学生来说，建议做为必拿分题。

复杂数据结果示例

```
const nestedStructure = {
  name: "John",
  age: 30,
  hobbies: ["reading", "coding"],
  address: {
    street: "123 Main St",
    city: "Anytown",
    country: "USA"
  },
  friends: [
    {
      name: "Alice",
      age: 28,
      hobbies: ["painting", "traveling"],
      address: {
        street: "456 Park Ave",
        city: "Otherville",
        country: "USA"
      }
    },
    {
      name: "Bob",
      age: 32,
      hobbies: ["gardening", "cooking"],
      address: {
        street: "789 Elm St",
        city: "Smalltown",
        country: "USA"
      }
    }
  ]
};

console.log(nestedStructure);
```

7. 其他需要掌握的知识点：

Math.floor() Math.ceil() Math.round() Math.random() Math.max() Math.min() parseFloat() parseInt()

8. 日期函数 Date 的基本使用

备考建议： 函数封装题目占分比例较高，难度不等，对于简单的函数封装题，一定要争取拿到全部分数，对于复杂的函数封装题，可以根据自己的实际情况和做题时间，争取拿到一些分数。

函数封装题，看到题不要着急做题，先理清解题解题思路，然后再去下手，函数封装题主要考察学生的逻辑思维以及对基础知识的掌握程度，在加上细心程度。也是前端 js 部分考察的重点。