- 第一题

```
<body>
    <input type="text" id="input">
    <span id="span">
    </span>
    <script>
        const input = document.getElementById('input')
        const span = document.getElementById('span')
        input.onchange = function () {
            span.innerText = input.value
        }


    </script>
```

用 change 事件有个缺点，只有当输入框失去焦点后，才会触发事件，不是实时输入实时更新，可以考虑使用 input 事件

- 第二题

```
番茄游戏笔试题-2.html > html > body > script
    function commonParentNode(oNode1, oNode2) {
        /*因为DOM树是树形结构，我们可以计算oNode1和oNode2的深度，以document.body的结点为根节点，其深度为0
        */
        const root = document.body
        let node1Dep = getDep(root, oNode1)
        let node2Dep = getDep(root, oNode2)
        let parent
        if (node1Dep <= node2Dep) parent = getCommonParent(oNode1, node1Dep, oNode2, node2Dep)
        else parent = getCommonParent(oNode2, node2Dep, oNode1, node1Dep)
        console.log(oNode1, '和', oNode2, '的共同的父节点为', parent);
    }
    function getDep(root, node) {
        let cur = node
        let dep = 0

        while (cur != root) {
            dep++
            cur = cur.parentNode
        }
        return dep
    }
    function getCommonParent(node1, node1Dep, node2, node2Dep) {
        //默认node1的深度小于node2的深度
        count = node2Dep
        let parent1 = node1, parent2 = node2
        while (parent1 != parent2) {
            if (count != node1Dep) {
                //这里是让深度更深的结点找到其父节点 使得父节点和 深度浅一些的结点 深度一样
                parent2 = node2.parentNode
                count--
            } else {
                parent1 = parent1.parentNode
                parent2 = parent2.parentNode
            }
        }
        //return parent1 和 parent2 都是一样的
        return parent1
    }
```

测试数据

```
<body>
    <div class="root">
        <div class="father1">
            <div id="son1-1"></div>
            <div id="son1-2"></div>
            <div id="son1-3"></div>
        </div>
        <div class="father2">
            <div id="son2-1">
                <div id="baby"></div>
            </div>
            <div id="son2-2"></div>
            <div id="son2-3"></div>
        </div>
    </div>
```

```
//测试实例
const son2_1 = document.getElementById('son2-1')
const baby = document.getElementById('baby')
commonParentNode(son2_1, baby)
const son2_2 = document.getElementById('son2-2')
commonParentNode(son2_1, son2_2)
const son1_1 = document.getElementById('son1-1')
commonParentNode(son1_1, son2_1)
```

测试输出结果

```
<div id="son2-1">…</div>  '和'  <div id="baby"></div>  '的共同的父节点为'  <div id="son2-1">…</div>
<div id="son2-1">…</div>  '和'  <div id="son2-2"></div>  '的共同的父节点为'  <div class="father2">…</div>
<div id="son1-1"></div>  '和'  <div id="son2-1">…</div>  '的共同的父节点为'  <div class="root">…</div>
>
```

- 第三题

```
const _rank = arr => {
    arr.sort((a, b) => {
        let sum1 = 0, sum2 = 0
        for (let key in a) sum1 += a[key]
        for (let key in b) sum2 += b[key]
        return sum2 - sum1
    })
    return arr
}
```

测试数据

```
const arr = [
    { amount: 30, activeValue: 66, rank: 80 },
    { amount: 69, activeValue: 56, rank: 80 },
    { amount: 70, activeValue: 90, rank: 80 },
    { amount: 99, activeValue: 64, rank: 80 },
]
```

测试输出结果

```
console.log(_rank(arr));
```

```
▼(4) [{…}, {…}, {…}, {…}]
  ▶0: {amount: 99, activeValue: 64, rank: 80}
  ▶1: {amount: 70, activeValue: 90, rank: 80}
  ▶2: {amount: 69, activeValue: 56, rank: 80}
  ▶3: {amount: 30, activeValue: 66, rank: 80}
   length: 4
  ▶[[Prototype]]: Array(0)
```

- 第四题

```
const people = [
    { id: '番茄1号', age: '20岁' },
    { id: '番茄2号', age: '21岁' },
    { id: '番茄3号', age: '19岁' },
]
function render1() {
    //第一种方式:createElment
    const container = document.getElementById('container')
    people.forEach(i => {
        let son = document.createElement('div')
        son.innerText = `${i.id}  ${i.age} `
        container.appendChild(son)
    })
}
// render1()
function render2() {
    //第二种方式: innerHTML拼接字符串
    const container = document.getElementById('container')
    let str = ""
    people.forEach(i => str += `<div>${i.id}  ${i.age}</div> `)
    container.innerHTML = str
}
// render2()
```