

2018 阿里正式试题

1. 下面代码中，当点击 点我 时，输出的正确结果是：B

```
<div id="div2">

<div id="div1">点我</div>

</div>

var div2=document.getElementById('div2');

var div1=document.getElementById('div1');


div1.addEventListener('click', function(event){console.log("A");},
true);

div2.addEventListener('click', function(event){console.log("B");});

div1.addEventListener('click', function(event){console.log("C");},
false);

div2.addEventListener('click', function(event){console.log("D");},
true);
```

A、A B C D

B、D A C B

C、A D B C

D、D C A B

E、B D A C

2. 关于 Fetch API，以下描述错误的是：F

A、fetch() 返回的是一个 Promise 实例

B、Fetch API 可以结合 async / await 使用

C、Fetch API 提供的 API 囊括但不限于 XHR 的所有功能

D、Fetch API 可以跨域

E、Fetch 提供了对 Request 和 Response 对象的通用定义

F、fetch() 必须接受一个参数：资源的路径

3. 以下代码片段在 Node.js 环境下执行的结果顺序是：C

```
setTimeout(function () {  
  console.log('1');  
}, 0);  
  
process.nextTick(function() {  
  console.log("3");  
});  
  
console.log('2');
```

```
setImmediate(function() {  
  console.log("4");  
});
```

A、 2、 1、 4、 3

B、 2、 1、 3、 4

C、 2、 3、 1、 4

D、 4、 1、 2、 3

E、 4、 2、 1、 3

F、 1、 4、 2、 3

4. 以下哪个标签不属于 svg 元素： C

A、 <circle>

B、 <ellipse>

C、 <rectangle>

D、<marker>

E、<polyline>

F、<animate>

5.

关于 ES6 当中箭头函数说法错误的是:A

A、箭头函数可以使用 yield

B、箭头函数不可以用作构造函数

C、不会改变 this 的指向

D、箭头函数是 Function 的实例

E、箭头函数函数体没有花括号可以做为返回值

F、内部不能访问到 arguments 对象

6. 关于 JavaScript 中的函数，以下说法正确的有：BF

在已知名称的函数的函数体声明语句之外，不能获知该函数的形参个数

A、在函数内部，可以通过 `arguments` 获取函数的实参个数

B、因为 `arguments` 是 `Array` 的实例，因此可以使用数组的方法去操作它

C、对同一个函数 `foo`，使用 `new foo()` 和 `foo()` 调用的结果是一样的

D、如果一个函数中没有使用 `return` 语句，则它默认返回 `null`

E、如果函数的实参是一个引用对象，则对应形参会和该实参指向同一个对象

F、如果函数的实参是一个引用对象，则对应形参会和该实参指向同一个对象

7. 关于 CSS 的 `position` 属性，下列说法中正确的是：AD

默认值是 `relative`

A、值为 `static` 时，`left`、`right`、`top`、`bottom` 的值无效。

B、`fixed` 相对于页面视口定位

C、`absolute` 是相对于 `body` 标签定位

D、absolute 的元素可以设置外边距（margins），且不会与其他边距合并

E、fix 和 absolute 相对的定位对象是一样的

10. 关于 ES6 类（Class）的实现，以下表述正确的是：ABDE

A、ES6 的 class 只是一个语法糖，实际上还是基于原型来实现的

B、如果没在 class 里面定义 constructor 方法，编译器会自动帮你添加

C、ES6 的 class 中支持定义私有属性

D、和 ES5 一样，同一个类的所有实例共享一个原型对象

E、如果没有显式指定构造方法，则会添加默认的 constructor 方法

修改基类的原型，派生类实例的原型不会被修改

11. 变量 data 为树状结构，数据大小层次不固定，格式如下：

```
const data = [  
  
  {  
  
    "id": '1',  
  
    "children": [  
  
      {
```

```
"id": '1-1',  
  
"children": [],  
  
"value": "a-1",  
  
},  
  
{  
  
"id": '1-2',  
  
"children": [],  
  
"value": "a-2",  
  
},  
  
],  
  
"value": "a",  
  
},  
  
{  
  
"id": '2',  
  
"children": [  
  
{  
  
"id": '2-1',  
  
"children": [  
  
{  
  
"id": '2-1-1',  
  
"children": [],  
  
"value": "c-1",  
  
},  
  
],  
  
],
```

```
"value": "b-1",  
  
},  
  
],  
  
"value": "b",  
  
},  
  
{  
  
"id": '3',  
  
"children": [  
  
],  
  
"value": "c",  
  
},  
  
];
```

请实现个方法 transformData, 递归处理数据, 给所有的父节点 (children 不为空的) 添加一个字段 relateId, 值为当前第一个子节点 (children 为空的) id 的值。

如上面的数据经过处理后的结果为:

```
[  
  
{  
  
"id": "1",  
  
"children": [  
  
{  
  
"id": "1-1",  
  
"children": [],  
  
"value": "a-1"
```



```
},  
  
{  
  "id": "1-2",  
  "children": [],  
  "value": "a-2"  
}  
  
],  
  
  "value": "a",  
  "relateId": "1-1"  
},  
  
{  
  "id": "2",  
  "children": [  
    {  
      "id": "2-1",  
      "children": [  
        {  
          "id": "2-1-1",  
          "children": [],  
          "value": "c-1"  
        }  
      ],  
      "value": "b-1",  
      "relateId": "2-1-1"
```

```

    }

    ],

    "value": "b",

    "relateId": "2-1-1"

  },

  {

    "id": "3",

    "children": [],

    "value": "c"

  }

]

```

12. 下面 HTML 中的内嵌 JS 代码会生成一个列表，格式为 "{index}. {点击目标的全名}"。于此同时当点击列表中的某个名字会在控制台中输出 "click on no. {点击目标的 index} {firstName}, {lastName}"。请尝试指出代码中存在的 BUG 以及可能会发生的性能问题，并从优雅、高效、可读性以及性能方面进行优化，在指出问题的同时请尽量给出修正代码。

```
<meta charset="UTF-8">
```

```
<title>Title</title>
```

```
<ul id="list">
```

```
</ul>
```

```
<script>

maxLength = 4;

list = document.querySelector('#list');


function processName(name) {

return {

firstName: name[0],

lastName: name[1],

getFullName() {

return this.firstName + ' ' + this.lastName;

},

};

}


var names = [

['Gregor', 'Bachmann'],

['Anita', 'Bruns'],

['Anke', 'Dorn'],

['Ulrich', 'Koch'],

['Dietrich', 'Riedl'],

['Wolfgang', 'Jahn'],

['Gesine', 'Sturm'],

['Theodor', 'Petersen'],

];
```

```
var validCount = 0;

for (var i = 0; i < names.length; i += 1) {

var flag1 = names[i][0].indexOf('A') !== 0;

var getFullName;

if (flag1 && names[i][0].length >= 4) {

getFullName = processName(names[i]).getFullName;

var lastName = processName(names[i]).lastName;

var firstName = processName(names[i]).firstName;

var span = document.createElement('li');

var textNode = document.createTextNode(i + 1 + '. ' + getFullName());

span.appendChild(textNode);

span.addEventListener('click', function () {

console.log('click on no.' + i + ' ' + firstName + ', ' + lastName);

});

if (validCount + 1 > maxLength) {

continue;

}

validCount += 1;

list.appendChild(span);

}

}

</script>
```